

CURSO PRÁTICO

1

DE PROGRAMAÇÃO DE COMPUTADORES

# INFORMATICA

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA



Cz\$ 20,00



# INPUT

Vol. 1

Nº 1

## NESTE NÚMERO

### PROGRAMAÇÃO EM CÓDIGO DE MÁQUINA

## PROGRAMAÇÃO EM CÓDIGO DE MÁQUINA

Para programar jogos, é fundamental conhecer o código de máquina e, particularmente, a linguagem Assembler. .... 1

### PROGRAMAÇÃO DE JOGOS

## ANIMAÇÃO E SINAIS GRÁFICOS

Produza seus próprios "desenhos animados", trabalhando com caracteres gráficos. .... 3

### PROGRAMAÇÃO BASIC

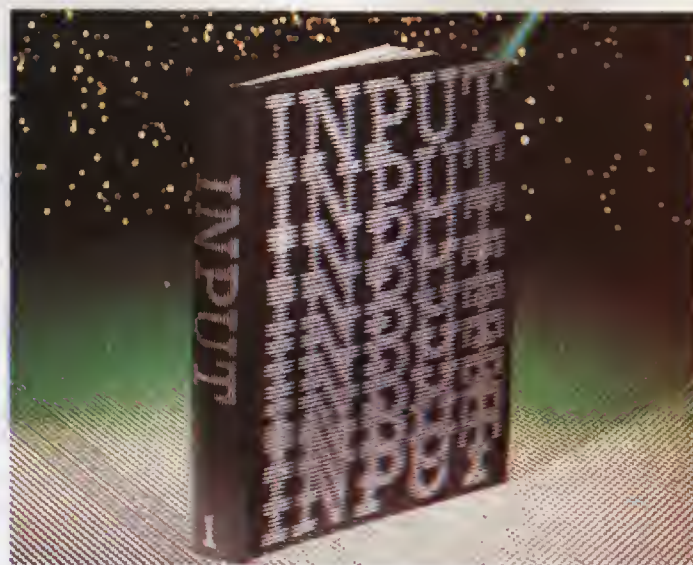
## NÚMEROS AO ACASO

Muitos programas de jogos são criados segundo a fórmula dos números aleatórios. Veja como isso acontece e aprenda a usar variáveis e a fazer comparações IF... THEN... .... 10

### APLICAÇÕES

## ESCREVA CARTAS SEM ESFORÇO

Como produzir cartas em série, ao mesmo tempo padronizadas e personalizadas. .... 17



## PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

## COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: **1. Pessoalmente** — por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em São Paulo os endereços são: Rua Brigadeiro Tobias, 773 (Centro); Av. Industrial, 117 (Santo André); e, no Rio de Janeiro: Rua da Passagem, 93 (Botafogo). **2. Por carta** — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidor Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132 (Jardim Tereza) — CEP 06000 — Osasco — São Paulo. **3. Por telex** — Utilize o nº (011) 33670 ABSA. Em Portugal, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Ltd. — Qta. Pau Varais, Azinhaga de Fetais — 2685, Camarate — Lisboa; Tel. 257-2542 — Apartado 57 — Telex 43 069 JARLIS P.

Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na Agência do Correio. **Atenção:** Após seis meses do encerramento da coleção, os pedidos serão atendidos, dependendo da disponibilidade de estoque. **Obs.:** Quando pedir livros, mencione sempre o título e/ou o autor da obra, além do número da edição.

## COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.



Editor

VICTOR CIVITA

### REDAÇÃO

Diretora Editorial: Iara Rodrigues

Editor chefe: Paulo de Almeida

Editor de texto: Cláudio A.V. Cavalcanti

Editor de Arte: Eduardo Barreto

Chefe de Arte: Carlos Luiz Batista

Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos,

José Maria de Oliveira, Grace A. Arruda,

Monica Lenardon Corradi

Secretária de Redação/Coordenadora: Stefania Crema

Secretários de Redação: Beatriz Hagström, José Benedito

de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares

de Andrade, Mauro de Queiroz

Secretário Gráfico: Antonio José Filho

### COLABORADORES

Consultor Editorial Responsável: Dr. Renato M.E. Sabbatini  
(Diretor do Núcleo de Informática Biomédica da  
Universidade Estadual de Campinas)

Execução Editorial: DATAQUEST Assessoria em Informática  
Ltda. Campinas, SP.

Tradução: Aluísio J. Dornellas de Barros, Maria Fernanda  
Sabbatini, Maristela G. Souza Machado

Adaptação, programação e redação: Aluísio J. Dornellas de  
Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli

Coordenação geral: Rejane Felizatti Sabbatini

Assistente de Arte: Dagmar Bastos Sampaio

### COMERCIAL

Diretor Comercial: Roberto Martins Silveira

Gerente Comercial: Flávio Ferruccio Maculan

Gerente de Circulação: Denise Maria Mozol

### PRODUÇÃO

Gerente de Produção: João Stungis

Coordenador de Impressão: Atilio Roberto Bonon

Preparador de Texto/Coordenador: Eliel Silveira Cunha

Preparadores de Texto: Ana Maria Dilguerian, Antonio  
Francelino de Oliveira, Karina Ap. V. Grechi, Levon Yacubian,  
Maria Teresa Galluzzi, Paulo Felipe Mendrone  
Revisor/Coordenador: José Maria de Assis  
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de  
Camargo, Lígia Aparecida Ricetto, Maria do Carmo Leme  
Monteiro, Maria Luíza Simões, Maria Teresa Martins Lopes

© Marshall Cavendish Limited, 1984/85.

© Editora Nova Cultural Ltda., São Paulo, Brasil,  
1986.

Edição organizada pela Editora Nova Cultural  
Ltda.

(Artigo 15 da Lei 5 988, de 14/12/1973).

Esta obra foi composta na AM Produções  
Gráficas Ltda. e impressa na Divisão Gráfica da  
Editora Abril S.A.

# PROGRAMAÇÃO EM CÓDIGO DE MÁQUINA

- O QUE É CÓDIGO DE MÁQUINA?
- VANTAGENS EM RELAÇÃO AO BASIC.
- COMO COMPREENDER CÓDIGO DE OPERAÇÃO E LINGUAGEM ASSEMBLER.

Utilizado na programação de jogos, o código de máquina proporciona uma ação rápida e contínua. Antes de empregá-lo, porém, você deve saber como ele afeta o desempenho do seu computador.

O BASIC constitui sem dúvida a mais difundida e popular linguagem de programação. Universalmente conhecido, ele é fácil de aprender e pode ser adaptado a diferentes máquinas. Seus programas, contudo, ocupam grandes espaços de memória e permitem apenas um movimento de cada vez. Assim, se um canhão atira, num jogo de guerra, o resto da ação tem que esperar, mesmo que seja por uma fração de segundo.

O BASIC usa palavras da linguagem humana, tiradas do inglês, e operações semelhantes às da aritmética, fáceis de compreender. Mas seu computador não pensa em inglês nem entende os símbolos aritméticos. Ele opera baseado em impulsos elétricos que representam números. E essa conversão é a causa de sua lentidão ao trabalhar com o BASIC.

Código de máquina é uma linguagem computacional composta apenas de números equivalentes àqueles que o computador utiliza. Assim, quando você emprega esse código não deve esperar que o computador responda em linguagem humana.

Consideremos um exemplo extraído dos computadores compatíveis com o Sinclair Spectrum; o código de máquina se parecerá com isto:

B9 28 08

Em BASIC o equivalente é:

100 IF A=C THEN GOTO 190

O código de máquina consiste, desse modo, numa série de números de dois dígitos. A letra B da linha acima é, na realidade, um símbolo. Ela representa o número 11, em notação hexadecimal.

Esses números hexadecimais são introduzidos na memória do computador. Instruções de operação, dados, números, letras, palavras e endereços de memória são representados por sinais de dois dígitos. E o computador identifica



a diferença entre essas informações pela ordem em que eles ocorrem no programa. Por exemplo, o primeiro número em qualquer programa precisa representar uma instrução. Se, por engano, você digitar um número qualquer nesse ponto, representando um dado ou um endereço de algum dado, o computador tentará interpretá-lo como uma instrução válida. A digitação desses números exige, portanto, uma precisão absoluta; caso contrário, o programa não funcionará.

## VANTAGENS DO CÓDIGO DE MÁQUINA

Quando você digita uma linha em BASIC, o computador tem que transpô-la para sua própria linguagem, antes de executá-la. Este é um processo trabalhoso que toma muito tempo, pois uma instrução em BASIC raramente é traduzida para um só comando ou declaração em código de máquina. Ela resulta, com frequência, em vários códigos de operação.

Quando se executa um programa completo escrito em BASIC, cada linha tem que ser interpretada seqüencialmente. O computador não armazena os resultados dessa tradução; se a mesma linha for executada novamente, o computador terá que interpretá-la outra vez.

Sempre que o computador encontra uma instrução em BASIC, executa as seguintes tarefas:

- reconhecer a instrução em BASIC;
- traduzir essa instrução para uma série de outras em código de máquina;
- executar as instruções, uma a uma;
- passar para a próxima linha do programa em BASIC.

Repetido muitas vezes, esse processo torna-se lento e moroso. É possível, porém, compilar um programa em BASIC, isto é, traduzi-lo integralmente para código de máquina, de uma só vez, antes de executá-lo, armazenando a tradução. A compilação é mais eficiente

que a interpretação, mas os programas resultantes são ainda bastante lentos.

Em contrapartida, a tradução do programa em código de máquina diretamente para a operação interna do computador quase não consome tempo. Nesse caso, há uma simples conversão de um número em outro, e não a tradução de um comando de linguagem em vários outros de código de máquina. Os programas tornam-se, assim, mais curtos e eficientes.

Os programas abaixo fazem a mesma coisa, usando BASIC e código de máquina. Compare suas velocidades relativas.



```
10 CLEAR 200,31000
20 DEFUSR0=31000
30 FOR N=31000 TO 31015
40 READ A
50 POKE N,A
60 NEXT
70 CLS0
80 PRINT @0,"ISTO ESTA EM BASIC"
```

```
90 FOR N=1 TO 500:NEXT
100 FOR N=1056 TO 1535
110 POKE N,PEEK(N+34000)
120 NEXT
130 FOR N=1 TO 1000:NEXT
140 CLS0
150 PRINT @0,"ISTO ESTA EM CODI
GO DE MAQUINA."
160 FOR N=1 TO 1000:NEXT
170 N=USR0(0)
180 FOR N=1 TO 2000:NEXT
190 DATA 206,136,240,142,4,32,1
66,192,167,128,140,6,0,38,247,5
7
```



Atenção: o programa abaixo está escrito em BASIC para computadores com o sistema DOS (disquete) e não será aceito pelo BASIC nível II normal (cassete).

Ao ligar a máquina, responda com o número 60000 à pergunta "Mem.usada?", ou ainda "Memory size?" (para computadores com 48 Kbytes de memória apenas).

```
10 CLS
20 DEFUSR0=-4536
30 FOR N=-4356 TO -4523
40 READ A
50 POKE N,A
60 NEXT N
70 CLS
80 PRINT "ISTO ESTA EM BASIC"
90 FOR I=1 TO 500:NEXT
100 FOR N=15360 TO 16383
110 POKE N,65
120 NEXT N
130 FOR N=1 TO 1000:NEXT
```

```
140 CLS
150 PRINT "ISTO ESTA' EM LIN-
GUAGEM DE MAQUINA"
160 FOR N=1 TO 1000:NEXT
170 N=USR0(0)
180 FOR I=1 TO 2000:NEXT
190 DATA 33,0,60,17,1,60,1,255,
3,54,65,237,176,201
```



```
10 CLEAR 29999
20 FOR n=30000 TO 30011
30 READ a
40 POKE n,a
50 NEXT n
60 PRINT "Isto esta em BASIC"
70 FOR n=16384 TO 22527
80 POKE n,PEEK (n-16384)
90 NEXT n
100 CLS
110 PRINT "Isto esta em Codigo
de Maquina"
120 PAUSE 100
130 RAND USR 30000
140 STOP
150 DATA 33,0,0,17,0,64,1,0,24
,237,176,201
```



```
10 CLEAR200,&H0FFF
20 DEFINT A-Z
30 AD=&HE000:DEFUSR=AD
40 FOR I=0TO10
50 READAS
60 POKE AD+I,VAL("&H"+AS)
70 NEXT
80 DATA0E,98,ED,6B,0,1,06,FF,ED
,B3,C9
90 CLS
100 PRINT "Isto esta em BASIC"
110 FOR I=1 TO 1000:NEXT
120 SCREEN 2
130 PSET (0,0)
140 FOR I=1 TO 6144
150 OUT 152,RND(1)*255
160 NEXT I
170 CLS
180 SCREEN 0
190 PRINT"Isto esta em código d
e máquina"
200 FOR I=0 TO 1000:NEXT
210 SCREEN 2
220 PSET(0,0)
230 FOR I=1 TO 25
240 A=USR(B)
250 NEXT
260 GOTO260
```



```
5 HGR : HOME : VTAB 24
10 FOR I = 800 TO 836
15 READ N
20 POKE I,N
25 NEXT
30 PRINT "Isto esta em BASIC"
35 FOR I = 1 TO 300: NEXT
40 FOR I = 8100 TO 14000
45 POKE I, RND (1) * 256
```

```
50 NEXT
55 TEXT : HGR : HOME : VTAB 24
: PRINT "Isto esta em linguagem
de maquina"
60 FOR I = 1 TO 3000: NEXT
65 CALL 800
70 DATA 169,0,133,20,133,22,1
69,32
80 DATA 133,21,169,193,133,23
,160,255
90 DATA 177,22,145,20,136,208
,249,165
100 DATA 21,201,63,208,1,96,2
30,21,230,23,76,46,3
```

Como você vai ver, tudo que o programa faz é encher a tela de caracteres ao acaso ("lixo", na gíria dos programadores). Entretanto, a velocidade com que a versão em linguagem de máquina faz isso é incomparavelmente maior do que a do programa em BASIC.

### LINGUAGEM ASSEMBLER

A grande dificuldade do código de máquina surge quando se quer escrevê-lo ou depurá-lo de erros. Poucas pessoas conseguem lembrar-se de todos os códigos numéricos e instruções. Para complicar ainda mais, os códigos de operação (opcodes) não são distinguíveis dos outros números que alimentam o computador. Assim, você não consegue entender um trecho de programa, a não ser que o acompanhe desde o começo — o que não ajuda quando se está procurando erros no programa.

Os códigos numéricos de operação, além disso, diferem consideravelmente entre si, conforme o microprocessador que é usado no computador, de modo que traduzir programas em código de máquina de um tipo de computador para outro pode ser bastante difícil.

Uma forma de contornar esses obstáculos é subir um pouco mais de nível, e escrever o programa em uma linguagem mais fácil de se utilizar do que o có-



digo de máquina. Essa linguagem é conhecida como Assembly, ou linguagem Assembler (que vem do inglês, montagem).

Em Assembler, os códigos operacionais são representados por abreviações mnemônicas (isto é, fáceis de lembrar). Por exemplo, a operação de carregar (load, em inglês) uma memória com um número, tem a abreviatura **LD**. Uma instrução de desvio (jump) pode ser chamada de **J**, **JP**, ou **JMP**, conforme a sintaxe do Assembler em uso. Com algum treino, é possível ler programas em Assembly tão facilmente quanto em BASIC, embora entendê-los seja algo mais complicado.

A desvantagem da linguagem Assembler é que o computador não pode utilizá-la diretamente, como no caso do código de máquina. Antes disso, o programa precisa ser montado por um Assembler, ou tradutor, que é um outro programa escrito em linguagem de máquina, ou mesmo em BASIC.

Mas o processo de montagem é bem mais simples, quando comparado com a interpretação de linhas em BASIC. A linguagem Assembler é equivalente ao código de máquina, ou seja, cada instrução em Assembly corresponde a uma instrução em código de máquina. Assim, a tradução do programa se processa palavra por palavra, número por número, etc. diretamente para código de máquina.

O Assembler também traduz o programa como um todo, antes que o computador o execute, ao invés de interpretá-lo linha por linha, enquanto

o programa está rodando. Isto dá vantagens adicionais de velocidade de execução.

Alguns computadores, como o Apple II, já vêm com um programa embutido, chamado monitor, que permite a entrada de códigos de máquina e dados em hexadecimal. Computadores com programas Assembler embutidos em sua ROM, entretanto, já são mais raros. No Brasil, o TK-2000, da Microdigital, é um exemplo. Para os outros computadores (linha TRS-80, TRS-Color, MSX, Sinclair Spectrum e ZX-81), existem programas Assembler disponíveis comercialmente, em fita ou disquete. Você poderá utilizar também, as versões de Assembler para cada máquina, que serão fornecidas mais adiante.

Mas, se você não tem acesso a um programa Assembler, pode fazer uma montagem manual. Mesmo para os programas mais simples, é mais fácil escrevê-los em Assembly, e depois traduzi-los manualmente para código de máquina em hexadecimal, usando as tabelas fornecidas com os manuais de programação Assembler.

Tudo isto pode parecer muito aborrecido; mas, antes de decidir que é isso mesmo, experimente esses programas em código de máquina. Eles são introduzidos via declarações **DATA** em programas BASIC. Os dados (**DATA**) estão listados no fim de cada programa.



```
10 CLEAR 200,31000
20 DEFUSR0=31000
30 FOR N=31000 TO 31020
40 READ A
50 POKE N,A
60 NEXT
70 N=USR0(0)
80 POKE 32767,RND(256)-1
90 FOR N=1 TO 100:NEXT
100 GOTO 70
110 DATA 142,4,0,206,136,184,16
120 DATA 6,192,184,127,255,138,129,167,1
130 DATA 28,140,6,0,38,242,57
```



As restrições na configuração da máquina para rodar o programa abaixo são as mesmas do primeiro programa.

```
20 DEFUSR0=-4536
30 FOR N=-4536 TO -4525
40 READ A
50 POKE N,A
60 NEXT N
70 N=USR(0)
80 POKE -4534,RND(20)
90 GOTO 70
100 DATA 33,0,0,17,0,60,1,0,4,
110 DATA 237,176,201
```



```
10 CLEAR 29999
20 FOR n=30000 TO 30020
30 READ a
40 POKE n,a
50 NEXT n
60 RAND USR 30000
70 GOTO 60
80 DATA 17,0,88,46,0,237,95,
90 DATA 71,58,140,92,128,230,63,103,1
100 DATA 0,3,237,176,201
```



```
10 CLEAR200,&HFFFF
20 DEFINT A-Z
30 AD=&HE000:DEFUSR=AD
40 FOR I=0TO10
50 READAS
60 POKE AD+I,VAL("&H"+AS)
70 NEXT
80 DATA0E,98,ED,6B,0,1,06,FF,ED
90 DATA B3,C9
90 FOR I=0 TO 1000:NEXT
100 SCREEN 3
110 PSET(0,0)
120 A=USR(B)
130 GOTO 120
200 FOR I=0 TO 1000:NEXT
210 SCREEN 3
220 PSET(0,0)
240 A=USR(B)
260 GOTO 240
```



Um aviso para quem for utilizar o programa acima: o programa em linguagem de máquina altera alguns endereços importantes da memória de trabalho do micro. Por isso, é impossível listá-lo depois de rodar uma vez. Assim, você deve digitar o programa e gravá-lo em fita ou disquete, antes de rodá-lo.

Para rodar no TK-2000, é necessário fazer as seguintes modificações: todos os valores 17, que aparecem dentro das declarações **DATA**, devem ser mudados para 20, e todos os valores 18, para 21.

Deste modo ele não se autodestrói.

```
10 FOR I = 800 TO 840
20 READ N
30 POKE I,N
40 NEXT
50 GR
60 CALL 800
70 DATA 169,0,133,17,169,193,
80 DATA 133,18,160
90 DATA 40,162,40,177,17,41,1
100 DATA 5,32,100
110 DATA 248,138,32,0,248,136,
120 DATA 208,242,202
130 DATA 208,239,169,254,197,
140 DATA 18,208,1,96,230,18,76,40,3
```

Agora, apenas para comparar, tente escrever um programa similar em BASIC. Acreditamos que você entenderá nossas razões. E mais tarde, veremos o que acontece aqui.



# ANIMAÇÃO E SINAIS GRÁFICOS

Programar jogos pode se revelar uma atividade fascinante. Mas não é certamente das mais fáceis. Por isso, você deve começar com coisas simples e ir avançando pouco a pouco. Esse procedimento lhe ensinará a pensar de maneira lógica, melhorando suas qualidades de programador. É o que vamos aprender nesta série de lições, até chegar ao desenvolvimento de jogos profissionais e complexos.

A primeira coisa que você precisa aprender para programar jogos, afora a linguagem BASIC, é a técnica de animação, pois a maioria dos jogos mais interessantes é do tipo "videogame", isto é, tem algum tipo de ação.

Para criar a ilusão de movimento, o programador usa praticamente as mesmas técnicas empregadas na elaboração de desenhos animados. Ele cria duas ou mais imagens e as alterna rapidamente — cerca de 24 vezes por segundo.

Existe, porém, uma diferença importante. Na animação de um desenho, o projetor é responsável pela eliminação das imagens desnecessárias. O mesmo não acontece com a animação por computador: neste caso, qualquer segmento de um desenho que você "projete" permanecerá na tela, a não ser que você imprima alguma coisa sobre ele, pois o computador não pode colocar, simultaneamente, duas imagens na mesma posição.

Por exemplo, se a linha 10 diz ao computador para colocar um *A* em um determinado lugar, qualquer linha impressa posteriormente — digamos um *B* —, no mesmo local, apagará o *A*.

Mas, e se não houver nada que você queira imprimir no lugar do caractere indesejado? Então você deve se lembrar de incluir no seu programa uma linha que coloque um espaço em branco na posição de interesse. Caso isso não seja feito, seu vídeo logo estará repleto de incômodos pedaços de braços, pernas e corpos!

Há grandes diferenças entre os computadores na maneira de se obter os caracteres (sinais) gráficos na tela. Os caracteres gráficos padronizados variam muito em tipo e disposição, assim como a maneira como são impressos na tela, e como são movimentados.

Uma animação muito simples, que ilustra os princípios da produção de efeitos de movimentação na tela do micro, é a do pequeno "inseto rastejante" (veja abaixo). Nas páginas seguintes mostraremos como conseguir esse efeito para diversos tipos de computadores.

```

  )))  )))  )))  )))
 000 < 000 < 000 < 000 <
  )))  )))  )))  )))

```



Para criar a figura do inseto na tela, programe o TRS-Color ou compatível (por exemplo, o Prológica CP-400), como segue abaixo. A declaração **PRINT @** (pronuncia-se "print arroba" ou "print em") é a maneira que o BASIC do Color tem para exibir um ou vários caracteres em um ponto específico da tela. O número que se segue após o sinal **@** é a posição na tela, que começa de 0, no canto superior direito, e aumenta de um em um da esquerda para direita e de cima para baixo. Como a tela do Color tem apenas 32 colunas, a primeira linha tem posições **@** indo de 0 a 31. A posição 32 já se situa na primeira coluna da segunda linha e assim por diante.

```

5 CLS
10 PRINT @238,"000"
20 PRINT @206,")))"
30 PRINT @241,"<"
40 PRINT @270,")))"

```

Este programa é uma forma bastante elaborada de criar uma imagem muito simples, mas ele ilustra alguns pontos interessantes:

Em primeiro lugar, ele lhe dá uma idéia a respeito das posições relativas na tela. O meio do inseto está, aproximadamente, no centro do vídeo, na localização 239. Como você vê, estamos usando caracteres "normais" (isto é, disponíveis no próprio teclado padrão, como a letra O, o sinal de parênteses, etc.) para compor a figura do "bicho".

Além disso, ele mostra o que acontece quando fazemos com que o computador imprima mais que um caractere em uma única posição de tela — ele simplesmente vai em frente e coloca os próximos caracteres nas posições subseqüentes (de numeração maior). Isto ex-

Se você quer dar mais vida aos seus jogos, comece a trabalhar com os caracteres gráficos mais simples, que já vêm programados na memória do seu computador.

plica por que as "antenas" da linha 30 estão em 241. As posições 238-240 já estão ocupadas pelo corpo do inseto.

Existe uma maneira melhor de desenhar o inseto, que é condensar o programa acima em uma única linha. E, se você ainda não descobriu, o TRS-Color tem uma abreviação para **PRINT**: o caractere "?". Quando o programa é listado, a máquina mostrará **PRINT** ao invés de "?".

O programa simplificado fica assim:

```

20 PRINT @238,"000<":PRINT @206,
,")))":PRINT @270,")))"

```

Se você acrescentar mais duas linhas, obterá alguma animação:

```

  )))  )))  )))  )))
 000 < 000 < 000 < 000
  )))  )))  )))  )))
20 PRINT @238,"000<":PRINT @206,
,"(((":PRINT @270,"(((("
30 GOTO 10

```

A execução deste programa produz uma imagem tremida e pouco nítida que não se parece nem um pouco com animação. A razão disso é que as imagens estão sendo trocadas rápido demais. Se você inserir um laço **FOR...NEXT** no programa, ele fará com que o computador pare e conte, deixando a animação mais clara. Qualquer número pode ser usado com os laços **FOR...NEXT** (linhas 15 e 25) e não apenas 15. Outros números resultarão em uma espera maior ou menor.

Experimente adicionar estas linhas:

```

15 FOR L=1 TO 15
17 NEXT L
25 FOR L=1 TO 15
27 NEXT L

```

Agora você tem um inseto que espera incessantemente, um tanto sem razão; apesar disso, a tela nos mostra uma animação convincente.

## GRÁFICOS DE BAIXA RESOLUÇÃO

Você pode usar os caracteres gráficos de baixa resolução do seu computador para criar figuras animadas mais interessantes. O Manual do Usuário mostra quais são esses caracteres. Cada um deles tem um código (um valor de 128



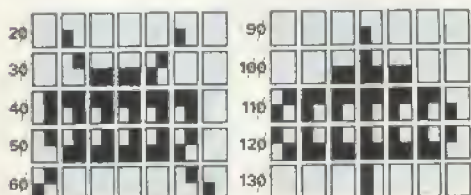
MOVIMENTO AS FIGURAS NO VIDEO  
COMO NUM DESENHO ANIMADO  
APRENDA A USAR  
CARACTERES GRÁFICOS



a 143 para os micros da linha TRS-Color). Para mostrá-los no video você usa a função **CHRS** seguida do código do caractere entre parênteses.

Por exemplo, para imprimir o caractere representado pelo código 138 no meio da tela, digite:

```
10 PRINT @ 239, CHR$(138)
```



Como construir o seu satélite.

O programa seguinte utiliza caracteres gráficos de baixa resolução para animar um satélite rotatório (a fig. 1 mostra os dois desenhos que são alternados no programa).

```
10 CLEAR 500:CLS
20 PRINT @174,CHR$(143)+CHR$(141):PRINT@178,CHR$(143)
30 PRINT @206,CHR$(140)+CHR$(132)+CHR$(141)
40 PRINT @236,CHR$(137)+CHR$(129)+CHR$(129)+CHR$(129)+CHR$(129)+CHR$(141)
50 PRINT @268,CHR$(135)+CHR$(132)+CHR$(132)+CHR$(132)+CHR$(133)+CHR$(135)
60 PRINT @300,CHR$(143):PRINT@303,CHR$(133):PRINT @305,CHR$(143)+CHR$(143)
70 FOR X=1 TO 20
80 NEXT X
90 PRINT @173,CHR$(141):PRINT@177,CHR$(141)
100 PRINT @205,CHR$(139)+CHR$(140)+CHR$(140)+CHR$(137)
110 PRINT @236,CHR$(138)+CHR$(130)+CHR$(130)+CHR$(130)+CHR$(130)+CHR$(130)+CHR$(143)
120 PRINT @268,CHR$(139)+CHR$(136)+CHR$(136)+CHR$(136)+CHR$(137)+CHR$(143)
```

```
130 PRINT @300,CHR$(137):PRINT@303,CHR$(143):PRINT @305,CHR$(139)+CHR$(141)
140 FOR X=1 TO 20
150 NEXT X
160 GOTO 10
```

Não se preocupe com a linha 10 do programa. **CLEAR 500** reserva espaço na memória para as cadeias de caracteres dos códigos **CHRS** e **CLS**. Examine os sinais gráficos representados pelos códigos após **CHRS** no programa e tente entender como a espaçonne foi construída. A instrução '+CHR\$( )' significa 'PRINT CHR\$( ) na próxima posição de tela'.

#### MOVIMENTO

Eis aqui o programa que anima o inseto, movendo-o pela tela:

```
10 CLS
20 FOR N=0 TO 28
30 PRINT @192+N,")):PRINT @224+N,"000<":PRINT @256+N,")):"
40 FOR X=1 TO 10
50 NEXT X
60 PRINT @192+N,"":PRINT @224+N,"":PRINT @256+N,""
70 PRINT @192+N,"{(":PRINT @224+N,"000<":PRINT @256+N,"{("
80 FOR X=1 TO 10
90 NEXT X
100 PRINT @192+N,"":PRINT @224+N,"":PRINT @256+N,""
110 NEXT N
120 GOTO 20
```

Existem três laços **FOR...NEXT** no programa. Os dois que utilizam **X** diminuem a rapidez de impressão fazendo o computador contar até dez; o que usa **N** faz com que o inseto se mova na tela.

Você pode não ter entendido por que a linha 20 mostra **FOR N = 0 TO 28** se existem 32 posições disponíveis em cada linha da tela. A razão para isso é que o inseto tem quatro espaços de comprimento, e, se houvesse mais que 28 na linha 20, a antena apareceria no lado oposto do video, uma linha abaixo. Isto acontece devido ao modo com que as posições de tela são numeradas. A posição 32, por exemplo, é a primeira da segunda linha, a partir do topo da tela.

As linhas 60 e 100 parecem não estar imprimindo nada; elas são as linhas "apagadoras" descritas anteriormente.

T

Para os micros da linha TRS-80 os programas acima podem ser modificados com facilidade, bastando alterar os números após os comandos **PRINT @**, já que nesses micros uma linha tem 64 colunas, ao invés das 32 da linha Color.

O meio da tela fica mais ou menos na posição 450. Subtraindo 64 dessa posição temos 386 para a posição inicial das perninhas de cima, e somando 64, obtemos 514 (para as perninhas de baixo). Finalmente, 453 nos dá a posição das antenas. O programa final fica assim:

```
10 PRINT @450,"000<":PRINT@386,
  ")))":PRINT@514,")))"
15 FOR L=1 TO 15
17 NEXT L
20 PRINT @450,"000<":PRINT@386,
  "(((":PRINT@514,"(((("
25 FOR L=1 TO 15
27 NEXT L
30 GOTO 10
```

#### TRABALHE COM CARACTERES GRÁFICOS

Estude agora com atenção tudo o que está dito na seção sobre o uso de caracteres gráficos no TRS-Color. Ela se aplica exatamente aos micros da linha TRS-80, que têm apenas caracteres gráficos de baixa resolução.

Os truques para desenhar figuras e movimentá-las são iguais aos ensinados para o TRS-Color, ou seja, basicamente, o comando **PRINT @**, e a função **CHRS** usada em conjunto com os códigos numéricos dos caracteres gráficos.

As duas únicas diferenças são as seguintes:

— O conjunto de caracteres gráficos do TRS-80 é bem maior, pois cobre os códigos de 129 a 191. Cada caractere é a combinação de seis pontos em uma grade de três de altura por dois de largura (o TRS-Color possui caracteres gráficos em uma grade de  $2 \times 2$ , ou seja, com combinações de quatro pontos por caractere).

— Como dissemos acima, o número de posições @ na tela também é maior: 1024 (dezesesseis linhas de 64 colunas cada).

Os efeitos gráficos de baixa resolução no TRS-80 são potencialmente mais ricos e detalhados do que o Color (com apenas uma desvantagem para o primeiro: não podemos escolher a cor dos caracteres gráficos no TRS-80, que tem o vídeo monocromático).

Experimente fazer um programa igual ao do Color, para traçar o satélite (combinação de vários **CHRS**), e depois

					128	128	158	128	128
					152	131	190	176	128
					128	136	157	172	128
					128	160	133	170	144
					136	145	128	128	128

					128	128	187	128	128
					168	172	157	142	128
					130	174	145	128	128
					176	186	130	148	128
					129	128	128	181	128

O corredor em duas posições.

faça algo mais difícil: animar um corredor de maratona. A definição da figura do corredor em caracteres gráficos da linha TRS-80 pode ser vista na fig. 2.



## SS

Abaixo vemos como programar o pequeno "monstro rastejante" para qualquer micro da linha Sinclair (por exemplo, o TK-85 ou o TK-90X). Para começar, tente criá-lo numa posição estática:

```
10 PRINT AT 10,15;"000"
20 PRINT AT 9,15;")))"
30 PRINT AT 11,15;")))"
40 PRINT AT 10,18;"<"
```

O programa usa a declaração **PRINT AT**, que serve para colocar em algum ponto da tela um caractere ou conjunto de caracteres. Os números que se seguem à declaração **PRINT AT** representam o número da linha onde ficará a figura (que vai de 0 a 21), e o número da coluna (que vai de 0 a 31). Assim, **PRINT AT 10,6;"A"**, por exemplo, pede ao computador para escrever a letra **A** na posição definida pela linha 10 e pela coluna 6 da tela (um quadriculado imaginário, como em um jogo de 'batalha naval').

Ele mostra o efeito de ordenar ao computador que imprima mais que um caractere numa mesma posição da tela: ele simplesmente avança e imprime os caracteres seguintes nas posições vizinhas. Esta é a razão pela qual as "antenas" do inseto na linha 40 estão em 10,18. As localizações 10,15; 10,16 e

10,17 já estão ocupadas pelo corpo do inseto.

Uma maneira mais conveniente de compor o inseto consiste em condensar as instruções acima em uma única linha; desta forma (só vale para o TK-90X):

```
10 PRINT AT 10,15;"000<";AT 9,15;")))";AT 11,15;")))"
```

Adicione agora mais duas linhas e você terá animação:

```
20 PRINT AT 10,15;"000<";AT 9,15;"((( ";AT 11,15;"((( "
30 GOTO 10
```

Quando executar esse programa, você verá que ele produz uma imagem pouco nítida. Isso acontece porque as imagens são trocadas muito rapidamente.

A melhor maneira de desacelerar o processo é usar um laço **FOR...NEXT** que faz com que o computador conte até dez (ou qualquer outro número que você queira). Antes de imprimir a imagem seguinte. Experimente, então, adicionar estas linhas ao seu programa (no TK-85 use 2 ao invés de 10):

```
15 FOR L=1 TO 10
17 NEXT L
25 FOR M=1 TO 10
27 NEXT M
```

Você pode variar a duração da pausa simplesmente mudando '1 TO 10' pa-

ra '1 TO 5' ou '1 TO 20', por exemplo.

O inseto criado até agora pode parecer inútil e estúpido, pois esperneia como louco mas não se move. Mais adiante veremos como essa situação se modifica (seção MOVIMENTO).

## COMO ANIMAR UMA FIGURA

Uma animação mais interessante pode ser produzida utilizando os caracteres gráficos padrão da linha Sinclair. Temos um exemplo na fig. 2.

O programa completo é dado mais abaixo; no entanto, se você não está habituado aos símbolos gráficos, será interessante criar, inicialmente, uma figura estática, compondo uma linha de cada vez, deste modo:

```
1 PRINT AT 5,15;"O"
2 PRINT AT 6,14;"
```

... e assim por diante.

Encontrar esses caracteres gráficos é relativamente fácil. Para obter aqueles da linha 2 acima, por exemplo, tecle **CAPS SHIFT (SHIFT no TK-85)** e **9**, simultaneamente. Isso o coloca no modo gráfico indicado pelo **G** piscando na tela. Então, no TK-90X, pressione **CAPS SHIFT** e **6** juntos para obter uma versão invertida — preto no lugar do branco — do símbolo da tecla **6**; depois **CAPS SHIFT** e **8** e **6** sozinhos. No

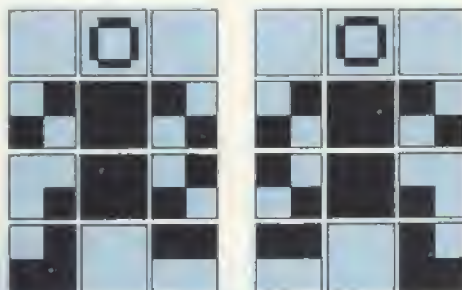


TK-85 pressione **SHIFT** e **T**; **SHIFT** e **SPACE**; **SHIFT** e **Y**. Finalmente, tecla 9 para deixar o modo gráfico antes de inserir as aspas no fim da linha.

Aqui temos o programa completo para formar a figura:

```
10 PRINT AT 5,14;"□□□";AT 6,
14;"■□■□";AT 7,14;"■□■□";
AT 8,14;"■□■□"
20 PRINT AT 5,14;"□□□";AT 6,
14;"■□■□";AT 7,14;"■□■□";
AT 8,14;"■□■□"
30 GOTO 10
```

É possível que você sinta necessidade de inserir novamente um laço **FOR...NEXT** depois da linha 10 e outro após a linha 20 para desacelerar a ação.



Um dançarino feito de doze quadradinhos.

## MOVIMENTO

Agora que você já sabe animar uma figura, podemos passar ao programa que faz com que ela se mova pelo vídeo:

```
10 FOR N=0 TO 27
20 PRINT AT 10,N;"OOO<";AT 9,
N;"((((";AT 11,N;"(((("
30 PRINT AT 10,N;" " ;AT 9,
N;" " ;AT 11,N;" " "
40 PRINT AT 10,N;"OOO<";AT 9,
N;")))";AT 11,N;")))"
50 PRINT AT 10,N;"OOO<";AT 9,
N;" " ;AT 11,N;" " "
60 NEXT N
70 GOTO 10
```

Este programa também usa um laço **FOR...NEXT**, porém com uma finalidade completamente diferente da do exemplo acima. Lá o computador fazia uma contagem, por frações de segundo, antes de imprimir uma imagem. Agora ele é responsável pela movimentação da imagem, que assume uma posição de cada vez na superfície da tela.

E por que a linha 10 mostra "O TO 27", se a tela do Sinclair tem, no total, 32 posições? Para descobrir, modifique a linha 10 para:

```
10 FOR N=0 TO 32
```

Outra dúvida pode ser em relação às linhas 30 e 50. Experimente apagá-las e você logo descobrirá sua utilidade.



O programa que movimenta a "tatu-rana" usa exatamente os mesmos caracteres que os outros computadores. O que muda é o comando para posicioná-lo, que nos computadores da linha MSX (por exemplo o HotBit, da Sharp) recebe o nome de **LOCATE**. Esse comando posiciona o cursor sobre a tela, em uma posição definida pelos dois números que se seguem ao comando **LOCATE**. O primeiro número refere-se à coluna, e o segundo, à linha. Assim, por exemplo, **LOCATE 12,4** coloca o cursor na coluna 12 e na linha 4. Qualquer comando **PRINT** que seja dado após isto escreverá o caractere (ou caracteres) a partir da posição definida pelo **LOCATE**. Agora, digite o programa:

```
5 CLS
10 LOCATE 18,10:PRINT "OOO"
20 LOCATE 18,9:PRINT ")))"
30 LOCATE 21,10:PRINT "<"
40 LOCATE 18,11:PRINT ")))"
42 FOR I=1 TO 30
47 NEXT I
50 LOCATE 18,10:PRINT "OOO"
60 LOCATE 18,9:PRINT "(((("
70 LOCATE 21,10:PRINT "<"
80 LOCATE 18,11:PRINT "(((("
82 FOR I=1 TO 30
87 NEXT I
90 GOTO 10
```

Quando o programa é executado, você pode observar a figura do "inseto"



vibrando rapidamente. Ela é criada pela sobreposição dos dois conjuntos de caracteres, que estão nas linhas com o comando **PRINT**. Ao mesmo tempo, o comando **GOTO**, na última linha, faz com que o programa se repita indefinidamente, pois retorna sempre ao seu início.

O movimento das patinhas da "taturana" é bastante veloz, e quase não dá para perceber o efeito da animação. Por isto, precisamos dar um jeito para diminuir a velocidade. A maneira mais fácil de se fazer isto é utilizarmos a declaração **FOR...NEXT**, que cria um laço de repetição 'no vazio', isto é, apenas para gastar um pouco de tempo (explicaremos na próxima lição de programação BASIC como se usa este tipo de laço).

Assim, introduza a linha:

```
45 FOR T=1 TO 100 : NEXT
```

A seguir, rode o programa e observe como o efeito de movimentação das patas tornou-se muito mais lento. Isso foi possível porque o laço criado na linha 45 funciona como um contador — neste caso, contando até 100, quando então, ao terminar a contagem, prosseguirá executando o programa a partir da linha 50.

Pode-se variar a duração dessa pausa simplesmente mudando-se o número 100 para outro qualquer. Quanto maior for esse valor numérico, maior será o retardo de tempo provocado.

### IMAGENS EM MOVIMENTO

O próximo passo é alterar o programa de forma que o corpo do inseto se movimente pela tela. Para isto, utilizamos a declaração **LOCATE**, já descrita.

Neste caso, usaremos uma variável para fazer com que a função **LOCATE** mude continuamente a posição do cursor, de modo que os **PRINT**'s que produzem a imagem dêem a impressão de deslocamento. Para fazer variar o valor desta variável que afeta o **LOCATE**, usamos um outro laço **FOR...NEXT** mais externo àquele que anima o bichinho:

```
10 FOR N=0 TO 36
20 LOCATE N,15:PRINT "ooo<"
30 LOCATE N,14:PRINT "(((("
40 LOCATE N,16:PRINT "(((("
50 LOCATE N,15:PRINT " " "
60 LOCATE N,14:PRINT " " "
70 LOCATE N,16:PRINT " " "
80 LOCATE N,15:PRINT "ooo<"
90 LOCATE N,14:PRINT ")))"
100 LOCATE N,16:PRINT ")))"
110 LOCATE N,15:PRINT " " "
120 LOCATE N,14:PRINT " " "
130 LOCATE N,16:PRINT " " "
140 NEXT N
```

O que fizemos foi simplesmente trocar o comando **GOTO** na linha 90 do programa anterior pelos comandos **FOR...NEXT**, que aumentam de um em

um o valor da variável **P**, cada vez que o programa se repete. Como **P** é argumento da instrução **LOCATE** (indexando o número de linha, ou o eixo X), isto faz a figura se movimentar um passo para a direita, a cada ciclo do programa.

As linhas 50, 60, 70, 110, 120 e 130 apagam as posições antigas do inseto. Retire-as e veja o que acontece.

Quando você roda o programa, o inseto atravessa a tela e pára em sua margem direita. Para fazê-lo voltar à origem e começar o "passeio" novamente, experimente digitar a seguinte linha adicional ao programa acima:

```
160 GOTO 10
```

### COMO CHEGAR AOS CARACTERES GRÁFICOS

Os computadores da linha MSX tem um grande conjunto de caracteres gráficos à disposição do programador dotado de instinto exploratório. Eles são digitados a partir do próprio teclado, e podem ser usados para criar figuras e desenhos mais elaborados.

Para ter acesso a estes caracteres, você deve utilizar as teclas **GRAPH**, **CODE** e **SHIFT**, juntamente com as demais teclas do teclado. No manual do seu computador podem ser encontradas "mapas" do teclado, que indicam as te-



clas associadas a cada caractere gráfico existente.

Agora, tente digitar alguns caracteres gráficos. O símbolo de espadas do baralho, por exemplo, pode ser obtido pressionando-se as teclas **GRAPH** e **Ç** (cedilha). O símbolo do naipe de ouro pode ser obtido pressionando-se as teclas **SHIFT**, **GRAPH** e **Ç** simultaneamente.

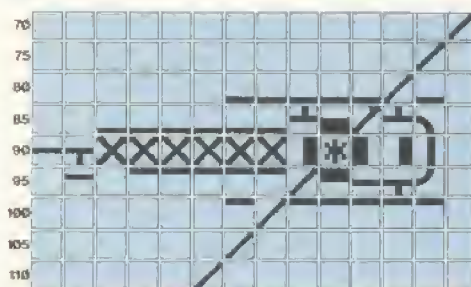
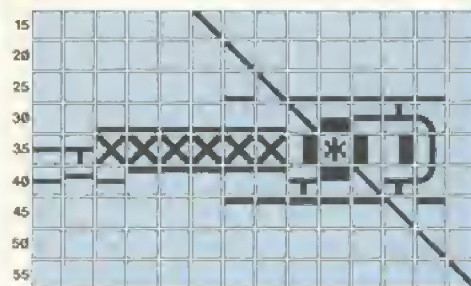
O programa abaixo faz a animação gráfica de um helicóptero visto de cima, demonstrando como podem ser usados os caracteres gráficos dos MSX.

```
5 CLS
10 LOCATE 13,5:PRINT
20 LOCATE 13,6:PRINT
30 LOCATE 13,7:PRINT
40 LOCATE 13,8:PRINT
50 LOCATE 13,9:PRINT
60 LOCATE 13,10:PRINT
70 LOCATE 13,11:PRINT
80 LOCATE 13,12:PRINT
90 LOCATE 13,13:PRINT
100 LOCATE 13,5:PRINT
110 LOCATE 13,6:PRINT
120 LOCATE 13,7:PRINT
130 LOCATE 13,8:PRINT
140 LOCATE 13,9:PRINT
150 LOCATE 13,10:PRINT
160 LOCATE 13,11:PRINT
170 LOCATE 13,12:PRINT
180 LOCATE 13,13:PRINT
190 GOTO 10
```



Os possuidores de micros compatíveis com a linha Apple II e derivados, bem como os que têm um Microdigital TK-2000, podem rodar os mesmos programas listados acima para os micros da linha MSX. Algumas modificações, porém, devem ser feitas, face às diferenças dos modelos:

- todos os comandos **CLS** (usados para limpar a tela) devem ser substituídos pelo comando equivalente **HOME**.



Como construir um helicóptero.

- nos micros da linha Apple não há a instrução **LOCATE**, mas ela pode ser substituída por duas outras, que são **HTAB** e **VTAB** (tabulação horizontal e vertical, respectivamente).

Essas duas funções controlam o posicionamento horizontal e vertical do cursor, antes de dar um **PRINT**. O comando **HTAB** é acompanhado de um número entre 1 e 40, que determina a coluna de posicionamento; já o comando **VTAB** faz o mesmo com a linha (de 1 a 21).

Assim, por exemplo, onde está escrito, no programa para o MSX:

```
20 HTAB 18:VTAB 10:PRINT "000"
```

substitua por:

```
20 LOCATE 18,10:PRINT "000"
```

O programa do helicóptero não pode ser introduzido em micros da linha Apple, pois estes não possuem caracteres gráficos. O mesmo não acontece com o TK-2000, cujos caracteres estão disponíveis através do teclado, ao se pressionar as teclas **<CONTROL>** e **B**, simultaneamente, ou por meio da função **CHRS (442)**, através de programa. Consulte o manual do TK-2000 e identifique os códigos dos caracteres usados nesse programa.

Se você quiser inventar novas figuras, basta usar os gráficos da tabela existente no manual ou os caracteres padronizados do teclado, ou os dois juntos. Caso seu computador seja diferente dos mencionados acima, proceda da seguinte maneira: desenhe as figuras em papel quadriculado e faça seu próprio programa, utilizando os caracteres gráficos disponíveis no seu computador.



# NÚMEROS AO ACASO

Pense em um número ao acaso: é assim, aleatoriamente, que muitos jogos são criados pelo computador.

Ninguém aprende a jogar futebol praticando apenas uma jogada de cada vez, para só entrar numa partida quando todos os lances possíveis estiverem assimilados. Se alguém escolhesse esse método, teria de esperar a vida inteira, pois o número de jogadas possíveis é praticamente infinito.

O mesmo, felizmente, não acontece com a programação de computadores, que pode ser aprendida de modo gradual. Assim, os manuais que acompanham os computadores ensinam uma função ou comando de cada vez. Você pode aprender por ele, se quiser. Mas a forma mais divertida de fazer isso é começar a jogar desde o começo, aprendendo na prática.

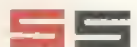
O jogo mais fácil de ser programado, em um micro doméstico é aquele em que o computador "inventa" um número aleatório (ao acaso), e o jogador tenta adivinhá-lo.

## A FUNÇÃO RND

Os computadores domésticos têm um gerador de números aleatórios (ou randômicos) que permite inventar jogos. Ele é operado, em BASIC, pela função **RND**. Em alguns computadores, entretanto, os números produzidos não são muito úteis na sua forma original — são todos frações decimais entre 0 e 0.99999999. Para verificar isso, digite este programa, teclando primeiramente **NEW** para limpar da memória qualquer programa já existente:



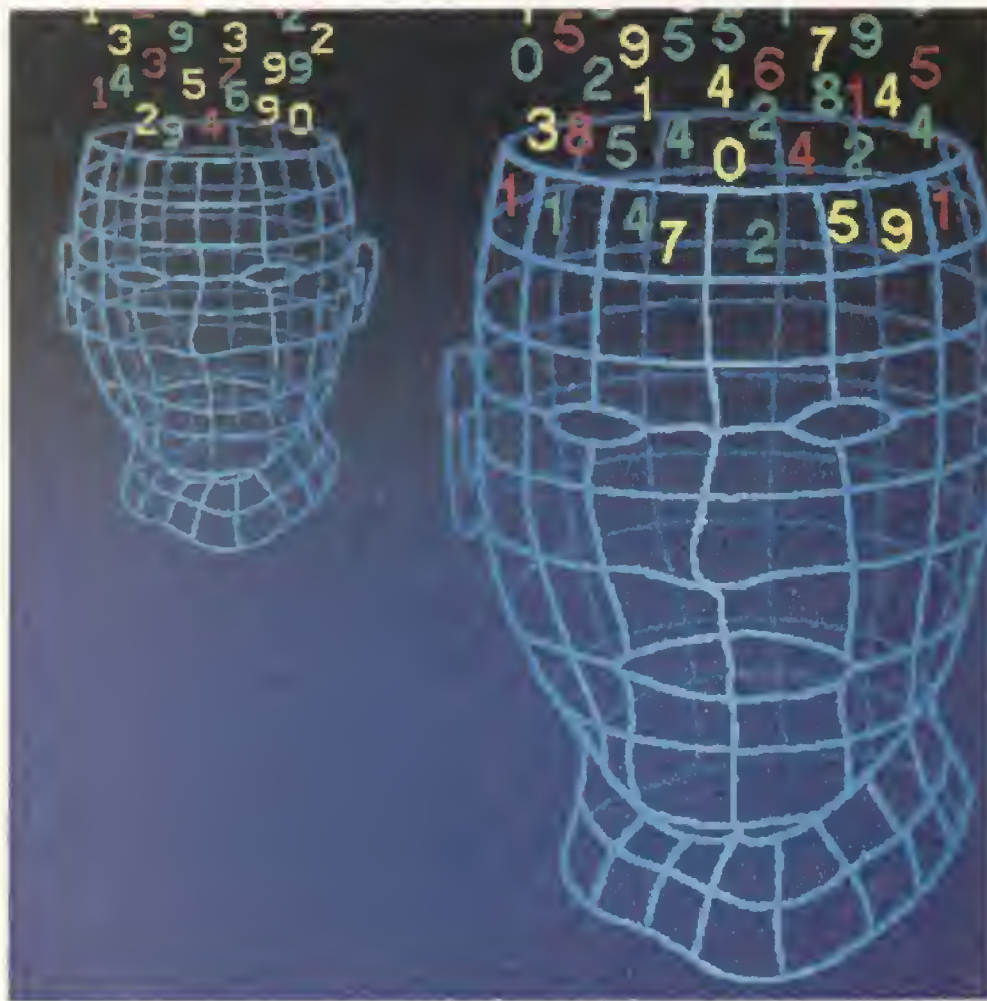
```
10 LET X=RND(0)
20 PRINT X
30 GOTO 10
```



```
10 LET X=RND
20 PRINT X
30 GOTO 10
```

- A FUNÇÃO RND
- COMO SÃO CRIADOS NÚMEROS ALEATÓRIOS
- APRENDA A USAR VÁRIAS
- A DECLARAÇÃO INPUT

- COMPARAÇÕES COM IF ... THEN
- DOIS JOGOS DE ADIVINHAÇÃO PARA VOCÊ PROGRAMAR
- FAIXAS DE GERAÇÃO DE NÚMEROS ALEATÓRIOS



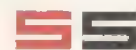
```
10 LET X = RND (1)
20 PRINT X
30 GOTO 10
```

(Lembre-se de teclar **<ENTER>** ou **<RETURN>** - dependendo do seu computador - após cada linha do programa.)

Quando você executar esse programa (use comando **RUN**), vai obter uma sequência de longos números decimais, muito distante de um jogo de adivinhação (para saber por que é assim, veja o quadro *Perguntas e Respostas* na página 13).

Então, como conseguir que o computador produza somente números inteiros? A resposta é muito simples: adicionando a função **INT** (uma abreviação da palavra inglesa *integer*, que quer dizer inteiro) a estas linhas.

nando a função **INT** (uma abreviação da palavra inglesa *integer*, que quer dizer inteiro) a estas linhas.



```
10 LET X=INT (RND*6)
```



```
10 LET X = INT ( RND (1) * 6)
20 PRINT X
30 GOTO 10
```

Isso vai gerar números inteiros entre 0 e 5. Nos computadores da linha TRS-80, não é preciso utilizar a função **INT**, pois a função **RND** a realiza se receber um número inteiro positivo como argumento:



```
10 LET X=RND(6)-1
20 PRINT X
30 GOTO 10
```

(Lembre-se sempre de pressionar <RETURN> ou <ENTER> após digitar cada linha.)

Qualquer que seja o computador que estiver usando, você não está limitado a trabalhar com números entre 0 e 5, podendo escolher igualmente 10 ou 10 000 como número máximo: o computador sempre sorteará um valor compreendido entre 0 e o número máximo escolhido.

### APRENDA A USAR VARIÁVEIS

Ao escrever o programa acima, além de selecionar um número aleatório, você deu um nome a ele (X). Daí em diante, no programa, toda vez que o X for colocado, o computador "saberá" que você se refere àquele número aleatório.

Esse nome, que permite ao computador identificar um valor, de tal forma que possa compará-lo com outro número, fazer operações aritméticas com ele, etc., é chamado de *variável*. A variável corresponde, grosso modo, ao rótulo de uma caixinha individual na memória do computador, onde armazenamos valores.

### A DECLARAÇÃO INPUT

Em nosso joguinho, após ter gerado o número aleatório, o passo seguinte será avisar o computador de modo a fazê-lo aceitar seu palpite. Para isso, você deve utilizar a declaração **INPUT**. Ela diz ao computador para ficar esperando até que alguma informação tenha sido digitada pela pessoa que está usando o programa.

A declaração **INPUT** sozinha, entretanto, não tem sentido. Precisamos fornecer ao computador um nome de variável para que ele saiba como identificar o dado a ser digitado e armazenar o seu valor na memória. Vamos supor que queremos entrar um dado na variável G (de *guess*, ou palpite, em inglês). Poderia ser perfeitamente um outro nome qualquer, envolvendo uma ou mais letras combinadas, tal como **PALPITE**, se você achar interessante.

Assim, a declaração completa fica da maneira como segue (não a digite ainda):



```
INPUT G
```

Agora que o computador tomou conhecimento do seu palpite, pode compará-lo com o número secreto gerado (que está guardado na variável X). Isso é feito de uma forma muito simples, equivalente à frase:

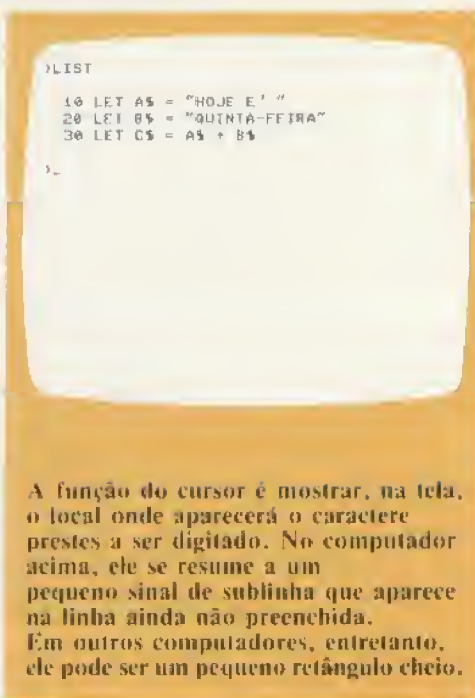
```
SE X=G ENTÃO ESCREVA "MUITO BEM!"
```

Em programação BASIC fica assim:



```
IF X=G THEN PRINT "MUITO BEM!"
```

A declaração **IF ... THEN**, sem dúvida, é muito útil. Você vai usá-la muitas vezes na programação.



A função do cursor é mostrar, na tela, o local onde aparecerá o caractere prestes a ser digitado. No computador acima, ele se resume a um pequeno sinal de sublinha que aparece na linha ainda não preenchida. Em outros computadores, entretanto, ele pode ser um pequeno retângulo cheio.

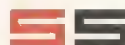
Os micros das linhas TRS-80, TRS-Color e MSX têm uma extensão a essa declaração que é o **IF ... THEN ... ELSE** (em bom português, **SE ... ENTÃO ... SENÃO**). Neste caso, o programador pede para o computador fazer alguma coisa a mais, caso a condição de teste (por exemplo, se os números são iguais) não seja satisfeita. Nos computadores que não têm a declaração **ELSE** — como é o caso dos compatíveis com a linha Apple e com as linhas Sinclair ZX-81 e Spectrum — se os dois números não forem iguais o programa passará automaticamente para a linha seguinte àquela onde está o comando **IF**.

### LISTAGEM DO PROGRAMA

Agora digite o programa abaixo (o sinal <> significa "é maior que e menor que" ou diferente de):

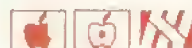


```
20 LET X=RND(6)-1
30 PRINT "O COMPUTADOR ESCOLHEU
UM NUMERO ENTRE 0 E 5. VOCE PO
DE ADIVINHA-LO?"
40 INPUT G
60 IF G=X THEN PRINT "MUITO BEM
!"ELSE PRINT "QUE AZAR - VOCE E
RROU!"
```



Para os micros compatíveis com o ZX-81, digite em maiúsculas.

```
20 LET X=INT (RND*6)
30 PRINT "O Computador escolheu
eu um numero entre 0 e 5. Tente
adivinha-lo."
40 INPUT G
60 IF G=X THEN PRINT "Muito
bem!"
80 IF G<>X THEN PRINT "Que
azar - Você errou!"
```



```
20 LET X=INT(RND(1)*6)
30 PRINT "O Computador escolheu
um numero entre 0 e 5. Tente a
divinhá-lo."
40 INPUT G
60 IF G=X THEN PRINT "Muito bem
!"
80 IF G<>X THEN PRINT "Que azar
- Você errou!"
```

Rode este programa e você verá que já dá para jogar, mas ainda não é muito divertido. Para começar, a tela vai ficando um tanto congestionada e, o que é pior, o jogo acaba logo após a primeira tentativa!

Para resolver o primeiro problema, você só precisa acrescentar:




```
10 CLS
50 CLS
```



```
10 HOME
50 HOME
```

A declaração **CLS** quer dizer "limpe a tela" (*clear screen*). Para os computadores da linha Apple, a declaração

## VARIÁVEIS ALFANUMÉRICAS



```

10 CLS
20 LET X=INT (RND*6)
30 PRINT "O Computador escolheu um numero entre 0 e 5. Tenha adivinha-lo."
40 INPUT G
50 CLS
60 IF G=X THEN PRINT "Muito bem!"
70 IF G=X THEN GOTO 90
80 IF G<>X THEN PRINT "Que azar - Voce errou!"
90 PRINT "Voce quer tentar outra vez? Se quiser, digite S e pressione ENTER"

```

```

91.15T
100 PRINT "I" ; "N" ; "P" ; "U" ; "T"
200 PRINT "I" ; "N" ; "P" ; "U" ; "T"
300 PRINT "I" ; "N" ; "P" ; "U" ; "T"
400 PRINT "I" ; "N" ; "P" ; "U" ; "T"
500 PRINT "I" ; "N" ; "P" ; "U" ; "T"

)RUN

INPUT

I
N
P
U
T

```



Mas, desta vez existe uma diferença importante. Depois da linha 20, o jogador deu entrada a um número. Agora ele vai entrar um S (para "sim") ou um N (para "não"), ou seja, uma letra não é um número.

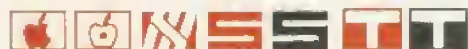
Isso significa que, na linha 100, ao invés de **INPUT A** você deverá usar **INPUT AS**. O cifrão (também chamado de "dólar"), designa o chamado **string** (cordão) e **AS** é uma variável alfanumérica (*string variable*).

Por que o \$ é necessário? Para entender isso é preciso conhecer a maneira como o computador armazena dados e trabalha com entradas, o que veremos mais adiante. Por enquanto, o importante é lembrar o seguinte: quando o computador deve esperar um número, você usa **INPUT A**, **INPUT B**, **INPUT X** ou qualquer outra letra; quando se trata de uma letra ou palavra, você deve usar **INPUT AS**, **INPUT BS**, **INPUT XS** etc.

A linha 120 foi incluída de forma que, se o jogador não deseja outra partida imediatamente, o computador espera até que ele queira, repetindo o processo até que a resposta seja igual a S. Isto acontece devido ao retorno constante à linha 100 até que a tecla S seja pressionada, quebrando o ciclo.

#### VOCÊ SABE TABUADA?

A função **RND** tem centenas de usos em programação. Imagine, por exemplo, que você deseje ensinar ao seu filho ou irmão a tabuada do nove. Você poderia fazer o seguinte:

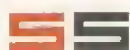


```
10 PRINT "QUANTO E 1 VEZES 9?"
20 INPUT A
30 IF A=9 THEN PRINT "CORRETO!"
40 PRINT "QUANTO E 2 VEZES 9?"
50 INPUT B
60 IF B=18 THEN PRINT "CORRETO!"
```

Mas, desse jeito, você teria um programa muito longo que não resolveria nem mesmo o problema de como proceder se uma das respostas estiver errada! Devemos usar a função **RND**, então, para produzir um programa que seja mais compacto e faça as perguntas corretamente, em sequência aleatória. Outra recomendação importante: trabalhe primeiro na parte principal do programa, deixando os enfeites para depois. Assim, experimente agora estas linhas (lembre-se de digitar o comando **NEW** antes!):



```
10 LET N=RND(12)
20 PRINT "QUANTO E ";N;" VEZES 9?"
30 INPUT A
40 IF A=N*9 THEN PRINT "CORRETO!"
```



Digite tudo o que vem abaixo apenas em maiúsculas, se for usar um computador com ZX-81:

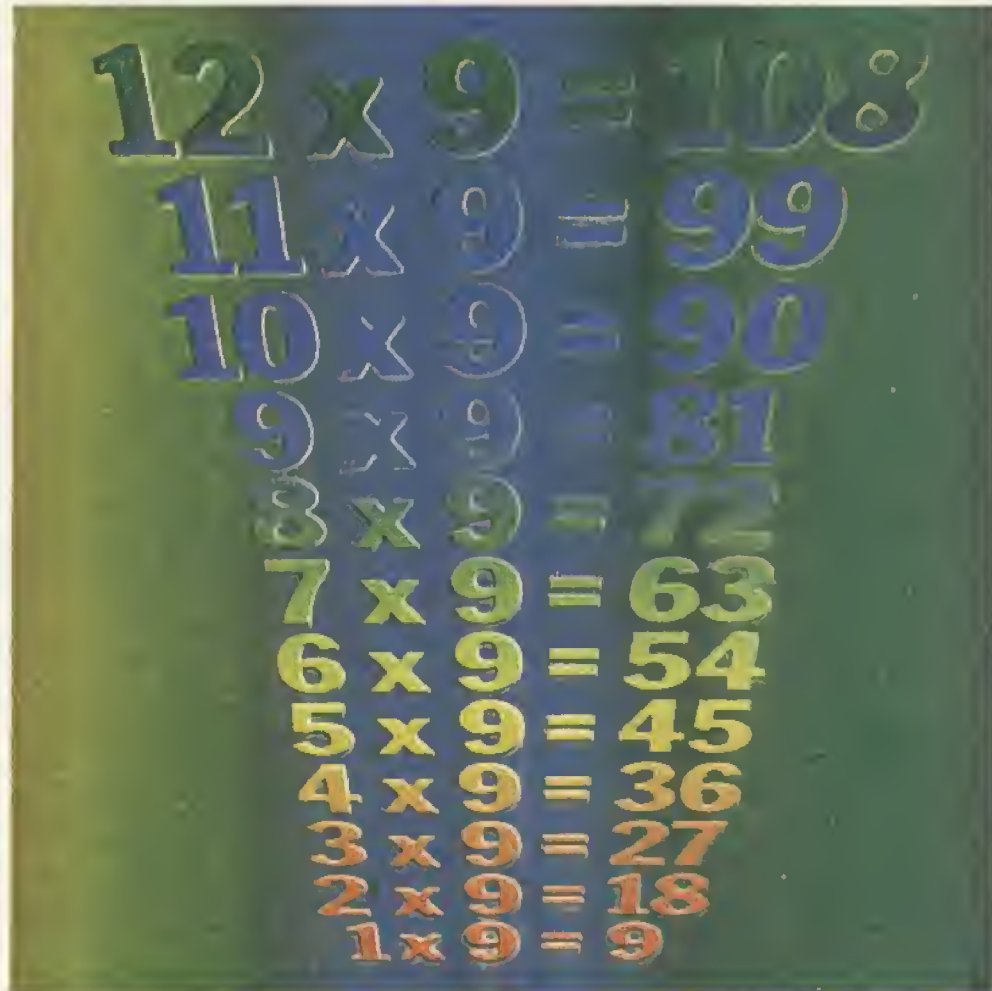
```
10 LET N=INT (RND*12+1)
20 PRINT "Quanto e ";N;" vezes 9?"
30 INPUT A
40 IF A=N*9 THEN PRINT "Correto!"
```



```
10 LET N = INT ( RND (1) * 12 + 1)
20 PRINT "QUANTO E ";N;"VEZES 9?"
30 INPUT A
40 IF A = N * 9 THEN PRINT "CERTO !"
```



```
10 LET N=INT(RND(-TIME)*12+1)
20 PRINT "Quanto é ";N;" vezes 9?"
30 INPUT A
40 IF A=N*9 THEN PRINT "Correto!"
```



```
>PRINT (1 + 2) * 3
3
>PRINT (4 * 5) / 3
1.33
>PRINT (8 - 6) ^ 2
4
>PRINT SQR (2)
1.41421356
```

Símbolos de computador utilizados em cálculos de aritmética elementar: o asterisco (\*), e não o x, é empregado para indicar multiplicação; a barra (/) significa "dividido por"; a flechinha para cima indica "elevado à potência de..." (em alguns computadores é um sinal circunflexo; em outros, dois asteriscos \*\*). Em compensação, os sinais que indicam as operações de soma e subtração são seus velhos conhecidos: o mais (+) e o menos (-) da aritmética elementar.

Note a função **TIME** sendo usada como argumento da função **RND**. Este é um artifício para gerar sempre números aleatórios diferentes nos micros da linha MSX, pois **TIME** informa ao computador o valor do tempo passado, em segundos, desde que o aparelho foi ligado.

Este programa usa o gerador **RND** de forma semelhante à do jogo de adivinhação. Na linha 10, determina-se uma variável para o número aleatório que o computador escolhe. Ela foi chamada de **N**, mas poderia ser outra letra ou palavra, desde que fosse usada sempre da mesma forma. Na parte direita da linha 10, o programa diz ao computador para escolher um número *inteiro* entre 1 e 12. Nos micros da linha Sinclair, é necessário somar 1, porque seus números aleatórios começam a partir de zero.

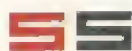
Na linha 20, pede-se ao jogador que multiplique por 9 o número que o computador escolheu desta vez. A linha 40 diz ao computador para multiplicar o número aleatório por 9 e comparar o resultado com a resposta dada pelo jogador (ou aluno). Se esta estiver correta, o computador mandará a mensagem de "CORRETO!" para a tela. Execute o programa e faça-o rodar várias vezes, para ver se funciona. Para que ele repita automaticamente as questões, acrescente a linha:

```
50 GOTO 10
```

Mas, pensando melhor, por que não fazemos as coisas de uma forma mais elegante, como se segue?



```
10 PRINT "OI. QUAL E O SEU NOME
?"
20 INPUT AS
30 CLS
40 PRINT "OLA. ";AS:PRINT"EU TE
NHO ALGUMAS PERGUNTAS PARA VOCE
"
50 FOR X=1 TO 3000:NEXT X
60 CLS
70 LET N=RND(12)
80 PRINT "QUANTO E ";N;" VEZES
9?"
90 INPUT A
100 IF A=N*9 THEN GOTO 150
110 CLS
120 PRINT A;"?"
130 PRINT "TENTE OUTRA VEZ."
140 GOTO 80
150 PRINT "MUITO BEM. ";AS;"!":
PRINT"AQUI VAI MAIS UMA:"
160 FOR X=1 TO 2000:NEXT X
170 GOTO 60
```



Digite tudo o que vem abaixo apenas em maiúsculas, se for usar um compatível com ZX-81:

```
10 PRINT "Oi. Qual e seu nome
?"
20 INPUT AS
30 CLS
40 PRINT "Ola. ";AS;"", "Eu t
enho algumas perguntas para v
oce."
50 PAUSE 200
60 CLS
70 LET N=INT (RND*12)+1
80 PRINT "Quanto e ";N;" veze
s 9?"
```

```
90 INPUT A
100 IF A=N*9 THEN GOTO 150
110 CLS
120 PRINT A;" ?"
130 PRINT "Tente outra vez!"
140 GOTO 80
150 PRINT "Muito bem, ";AS;"."
,"Aqui vai mais uma:"
160 PAUSE 150
170 GOTO 60
```



```
10 PRINT "Ola.Qual e o seu nom
e?"
20 INPUT AS
30 HOME
40 PRINT "Oi ";AS: PRINT "Eu t
```

```

enho algumas perguntas para voc
e."
50 FOR X = 1 TO 6000: NEXT X
60 HOME
70 LET N = INT ( RND (1) * 10
+ 1)
80 PRINT "Quanto e ";N;" vezes
9?"
90 INPUT A
100 IF A = N * 9 THEN GOTO 15
0
110 HOME
120 PRINT A;"?"
130 PRINT "Errado. Tente outra
vez"
140 GOTO 80
150 PRINT "Muito bem ";AS: PRI
NT "Aqui vai outra."
160 FOR X = 1 TO 4000: NEXT X
170 GOTO 60

```



```

10 PRINT "Oi. Qual é o seu nome
?"
20 INPUT AS
30 CLS
40 PRINT "Olá, ";AS;"", "Eu ten
ho algumas perguntas para você:
"
50 FOR X=1 TO 2500:NEXT X
60 CLS
70 LET N=INT(RND(-TIME)*12)+1
80 PRINT "Quanto é ";N;" vezes
9?"
90 INPUT A
100 IF A=N*9 THEN GOTO 150
110 CLS
120 PRINT A;"?"
130 PRINT "Tente outra vez."
140 GOTO 80
150 PRINT "Muito bem, ";AS;"!",
"Aqui vai outra:"
160 FOR X=1 TO 1500:NEXT X
170 GOTO 60

```

As linhas 30, 60 e 110 evitam que a tela fique repleta de mensagens do computador ou de respostas erradas. As linhas 50 e 160 fazem o computador esperar algum tempo antes de formular a próxima pergunta. As declarações **FOR...NEXT** serão explicadas na próxima lição de BASIC.

No programa para o Sinclair, as vírgulas nas linhas 40 e 150 servem para espaçar as mensagens na tela. Nos outros programas, as declarações **PRINT** extras fazem o mesmo. Para evitar que o programa seja interminável, faça o seguinte:

LIST

```

10 LET AS = "HOJE E "
20 LET BS = "QUINTA-FEIRA"
30 LET CS = AS + BS

```

```

PRINT
HOJE E " QUINTA-FEIRA
NEW
CLS

```

Os números colocados no começo de cada linha do programa são muito importantes. Sem eles, o computador executará cada linha diretamente e de forma separada, em vez de seguir o programa como um todo. Mesmo que elas sejam introduzidas fora de ordem, o computador as colocará na sequência certa (ou seja, em ordem ascendente), orientando-se pelos números de linha. Intervalos entre os números são utilizados para permitir a introdução de novas linhas, caso isso seja necessário.



Pressione a tecla **BREAK**



Acione as teclas **STOP** e **ENTER**.



Pressione as teclas **CONTROL** e **C**, simultaneamente, e depois **RETURN**.



Pressione as teclas **CONTROL** e **STOP**, simultaneamente. Mudando os nove do programa para cinco, seis ou sete, você poderá testá-lo com outras tabuadas. Será que você conseguiria modificar o programa para colocar esses números como variáveis, que seriam definidas logo no começo por outra declaração **INPUT**?

## MICRO DICAS

Muitas vezes, os iniciantes encontram dificuldades para interromper um programa em execução e voltar para a listagem. Isso acontece especialmente durante uma série de **INPUTs**, quando o computador enche a tela de mensagens, não importa o que você digite.

A seguir, abordaremos alguns pequenos truques que o ajudarão a sair desses aparentes impasses. Não existem problemas sem solução para quem lida com computadores.



Pressione simultaneamente as teclas **CONTROL** e **SPACE** (para acionar a função **BREAK**). Se isso não der resultado, é porque o programa está parado numa declaração. **INPUT**. Neste caso, se houver aspas na parte de baixo da tela, use a tecla de recuo do cursor e **DELETE**, para remover as aspas à esquerda. Depois disso, e se não houver aspas, pressione as teclas **STOP** e **ENTER**, e acione **ENTER** de novo, para listar o programa automaticamente.



Você deve acionar a tecla **CTRL**, mantê-la pressionada e, a seguir, pressionar a tecla **C**. Usaremos a notação **CTRL-C** para essa operação, assim como em qualquer outra ocasião em que duas teclas devam ser acionadas simultaneamente. **CTRL-C** não funciona experimentalmente **CTRL-RESET**. Após obter na tela o 'J', que é o sinal de prontidão, digite **LIST**.



Pressione a tecla **BREAK**, e depois digite o comando **LIST**. Se não funcionar, pressione a tecla **RESET** na parte de trás do computador.



Acione agora simultaneamente as teclas **CONTROL** e **STOP**.

### Especificação de faixas de números aleatórios

Números aleatórios entre 0 e 0.9999999  
 Números aleatórios entre 0 e N \* 0.9999999  
 Números aleatórios entre -10 e +10  
 Números aleatórios entre 0 e 39  
 Números aleatórios entre 1 e 40



RND (1)  
 RND (1)\*N  
 INT(RND (1) \*21)-10  
 INT(RND (1)\* 40)  
 INT(RND(1)\*40)+1



RND (0)  
 RND (0) \*N  
 RND (21)-11  
 INT(RND (0)\*40)  
 RND (40)



RND  
 N\*RND  
 INT(RND\* 21)-10  
 INT(RND\* 40)  
 INT(RND\*40)+1

# ESCREVA CARTAS SEM ESFORÇO

- COMO ENTRAR O PROGRAMA
- APRENDA A ADAPTAR O PROGRAMA

- A SUA IMPRESSORA
- DIAGRAMAÇÃO DE UMA CARTA
- PADRÃO
- MELHORE SEU TEXTO
- IMPRESSÃO DE CÓPIAS

**Você comete erros quando escreve cartas importantes? Se isso acontece, aqui está um programa que permite produzir cartas padronizadas com cópias personalizadas para muitas pessoas.**

Escrever um grande número de cartas similares (por exemplo, pedidos de emprego, cartões de Natal, etc.) é uma tarefa longa e tediosa, quando realizada manualmente.

O programa apresentado neste artigo resolve num instante esse tipo de problema. Ele é um modelo simplificado de um editor de textos, que permite entrar com uma carta padronizada no computador e produzir, caso necessário, cópias ligeiramente diferentes entre si. Alguns computadores pessoais, como os da linha MSX, por exemplo, têm recursos para acentuação correta de textos em português, letras minúsculas e maiúsculas no teclado e na tela, etc. Já os computadores compatíveis com a linha Sinclair ZX-81 são bastante difíceis de

usar com processamento de textos, pois dispõem de pouquíssimos recursos para isto. Assim, não apresentamos aqui uma versão para máquinas desse tipo.

Ao contrário de um programa completo de processamento de textos, que costuma ser muito complexo e tomaria horas de digitação para ser colocado no computador, o programa aqui apresentado tem bem menos recursos. Por exemplo, não dá para ver na tela o aspecto final da carta a ser impressa. Mas não se preocupe, ele não é tão limitado assim: tem recursos para evitar a divisão de palavras ao final de uma linha, inserir uma linha em branco entre dois parágrafos, etc.

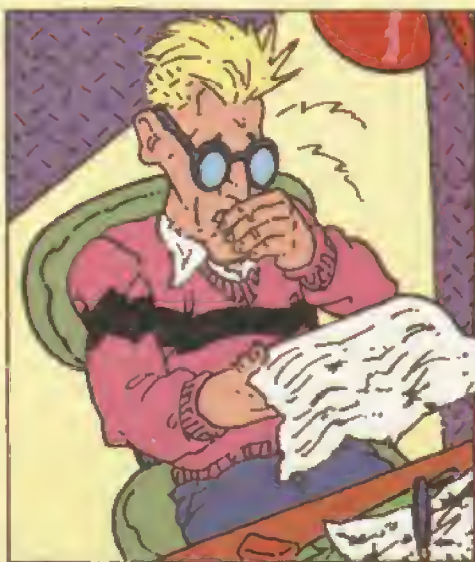
Qualquer tipo de impressora serve para produzir as cartas (lembre-se, entretanto, que se você usou acentuação no texto, é necessário que a sua impressora seja capaz de reproduzi-la, o que nem sempre acontece). Se você quiser uma melhor qualidade de impressão, como é o caso de cartas mais "profissionais", vai precisar, evidentemente, de uma impressora mais sofisticada (e com fita nova!). Esta pode ser do tipo matricial, com capacidade de imprimir em "qualidade car-

ta" (dupla sensibilidade de pontos), ou uma impressora (ou máquina de escrever) acoplada ao computador, usando o sistema de margarida, ou outro. Essas máquinas são caras, e nem todo mundo pode tê-las. Assim, talvez você possa usar a de um amigo ou de seu trabalho. As características das impressoras para micros serão discutidas em um artigo posterior.

## COMO "ENTRAR" O PROGRAMA

O programa consiste de duas partes: o programa propriamente dito, na primeira parte da listagem, e o texto da carta, que deve ser armazenado no programa, através de uma série de declarações **DATA**.

Comece digitando o programa principal, listado a seguir para cada tipo de máquina. Não digite as declarações **DATA** presentes no programa. Em seguida, grave o programa assim digitado em fita cassete ou disquete, usando o comando **SAVE** (em alguns computadores, esse comando chama-se **CSAVE**): consulte o manual do computador para ver como usá-lo. Assim, o programa gravado poderá ser usado para qualquer tipo de carta, depois. Se você quiser preservar a fita ou disco o programa contendo algum mo-



delo particular de carta, grave-o separadamente, com outro nome (programa mais dados).

Antes de imprimir sua carta, você deve fazer alguns ajustes, dependendo da impressora utilizada.

Em primeiro lugar, verifique quantos caracteres por linha sua impressora aceita (geralmente são 40, 80, 120 caracteres por linha). Então, modifique o programa de forma a seguir essa largura de linha, permitindo uma margem adequada de ambos os lados. No programa, a variável **TL** representa o comprimento total disponível na impressora, e **LL**, o comprimento da linha a ser digitada. Assim, você pode alterar os valores atribuídos no programa a essas variáveis, antes de imprimir as cartas.



Modifique a linha 20 (dando a **TL** o valor da largura de linha disponível na impressora) e a linha 30, de modo que o valor de **LL** contenha a largura da linha da carta. Do jeito que está no programa, a linha 40 mostra na tela como sairá a carta, antes de imprimi-la. No momento da impressão, modifique a variável **P**, nessa linha, de 0 para 2.



Modifique a linha 10 de maneira que **PL** seja a largura da impressora, e **LL**, a largura que você deseja para a carta (no exemplo dado, essas larguras estão muito pequenas para impressoras mais profissionais).



Nesse computador não dá para imprimir a carta primeiro no vídeo, antes de mandá-la para a impressora, a não ser que você modifique todas as declarações tipo **LPRINT** existentes, para **PRINT**, e vice-versa. Na linha 10, as variáveis **PL** e **LL** devem ser modificadas conforme a sua impressora (ou, para impressão em vídeo, para **LL=35** e **PL=40**).



Modifique a linha 30 de maneira que **TL** e **LL** contenham as larguras de linha disponíveis na impressora e na carta, respectivamente. Se você quiser ver na tela, primeiro, como sairá a carta, mude o valor de **P** para 0, **TL** e **LL** para 39.

#### DIGITE SUA PRIMEIRA CARTA

Sua carta consistirá de uma série de declarações **DATA**, uma linha de texto por declaração. Digite-as como mostrado no exemplo na última página deste artigo.

As instruções **DATA** devem começar na linha 1000 do programa, e conter inicialmente o seu endereço. A primeira tarefa do programa será procurar a linha mais longa do endereço, e imprimi-lo à direita do cabeçalho. Depois disso, as declarações **DATA** deverão conter o texto propriamente dito da carta. O computador alinhará esse texto junto à margem esquerda da carta, conforme os padrões mais modernos para cartas comerciais.

Cada linha da carta deve começar com aspas, logo após a declaração **DATA**. Alguns símbolos de edição podem ser incluídos no texto. Eles têm por objetivo orientar a forma estética de impressão. Aqui vai o significado de cada símbolo de edição para nosso programa:

Aqui vai o significado de cada símbolo de edição para nosso programa:

- O símbolo **#** significa: "esta linha é parte do meu endereço, coloque-a do lado direito da página".

- O cifrão, **\$**, quer dizer: "comece um novo parágrafo, deixando uma linha em branco acima dele".

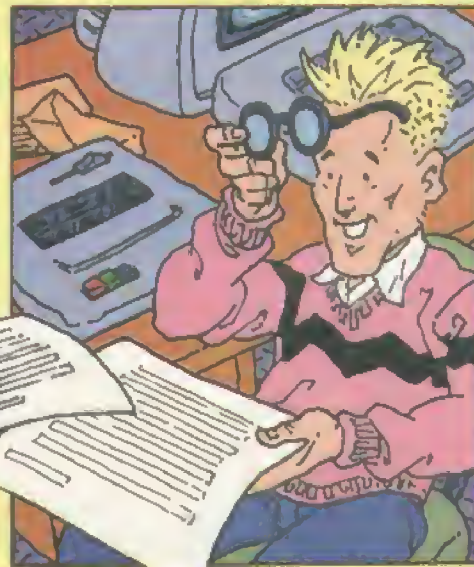
- O **&** comercial (*ampersand*): "comece uma nova linha na margem esquerda, mas não deixe uma linha em branco antes" (ele é útil para compor o endereço do destinatário da carta).

- O asterisco (**\***) significa: "centre esta linha".

Se você deseja colocar algumas linhas em branco indique-as com uma série de cifrões entre aspas, um para cada linha em branco. Quando você atingir o final da carta, os computadores do tipo abaixo necessitam de uma linha adicional:



Entre um número de linha adicional, seguido por **DATA S**. Entre outra linha adicional, seguida por **DATA**.



## MELHORANDO SUA CARTA

Visto que tudo está escrito em BASIC, você poderá, se quiser, gravar novamente o programa, com as declarações DATA incluídas.

Posteriormente, se você quiser melhorar a carta, ou produzir uma nova versão, o primeiro passo será carregar o programa correspondente, a partir do disco ou cassete (usando o comando LOAD, ou CLOAD, conforme o tipo de computador), e editar o programa usando o editor BASIC da forma habitual. Grave-a novamente, se quiser preservar essas mudanças. Caso contrário, terá que redigitá-las quando carregar o programa original novamente.

## IMPRIMINDO AS CARTAS

Para imprimir sua carta basta executar o programa: ele se encarregará de ativar e desativar a impressora. Se você tentar fazer isso sem que a impressora esteja conectada (quando ela está ligada, a luz que indica a conexão com o computador, normalmente rotulada LINHA ou LINE, permanece acesa), o computador ficará parado, esperando que a impressora entre em funcionamento. Neste caso, interrompa a execução do programa, pressionando a tecla BREAK ou sua equivalente. A forma de ver a carta no vídeo, antes de imprimi-la, já foi indicada.

O tipo de papel a ser usado depende de sua impressora. Algumas aceitam folhas soltas, caso tenham o rolo compressor de borracha: essas folhas são melhores para uma correspondência mais formal. A maioria das impressoras, entretanto, usa formulário contínuo, deslocado pelos picotes na margem (remalina). Depois de impressa a carta, retire as remalinas com uma guilhotina ou abrindo os picotes, acertando com uma tesoura ou lâmina afiada.

No caso de utilizar papel timbrado, não é necessário incluir o seu endereço na carta. Assim, digite o texto normalmente, começando da linha 1000, mas sem entrar as linhas que começam por #. Para imprimir cópias repetidas da mesma carta, execute o programa novamente, tantas vezes quantas forem necessárias.



```
10 CLEAR 200
20 TL=80
30 LL=56
```

```
40 P=0
50 SP=(TL-LL)/2:HW=TL/2
60 PRINT #P,CHR$(13)
70 IF P=2 THEN SP$=STRING$(SP,"
") ELSE SP=0:HW=16
200 READAS:AS=AS+" "
210 IF AS="" GOTO 290
220 OS=LEFT$(AS,1)
230 IF OS="#" THEN ML=0:GOSUB 4
00:GOTO 290
240 IF OS="$" THEN RL=0:PRINT#P,CHR$(13);CHR$(13);SP$;:ST=2:GOTO 280
250 IF OS="*" GOSUB 800:RL=HW-LEN(AS)/2:GOTO 280
260 IF OS="&" THEN RL=0:PRINT#P,CHR$(13);SP$;:ST=2:GOTO 280
270 ST=1
280 GOSUB 600
290 GOTO 200
400 IF LEN(AS)>ML THEN ML=LEN(AS)
410 N=N+1:READ AS:IF LEFT$(AS,1)="#" GOTO 400
420 IF ML>LL THEN CLS:PRINT "ERRO NO FORMATO. ENDEREÇO MUITO LONGO PARA O TAMANHO DE LINHA ESCOLHIDO." :END
430 RESTORE:FOR J=0 TO N-1:READ AS:PRINT#P,STRING$(LL-ML," ");SP$;:PRINT#P,MID$(AS,2);CHR$(13);
440 NEXT:RETURN
600 WL=0
610 IF ST+WL>LEN(AS) THEN
620 IF MID$(AS,ST+WL,1)<>" " THEN WL=WL+1:GOTO 610
630 IF WL>LL THEN CLS:PRINT "ERRO NO FORMATO..." :AS:PRINT "... CONTEM UMA PALAVRA GRANDE DEMASIADA!":END
640 IF RL+WL-1>LL THEN PRINT #P,CHR$(13);SP$;:RL=0
650 WL=WL+1
660 PRINT #P,MID$(AS,ST,WL):RL=RL+LEN(MID$(AS,ST,WL))
670 ST=ST+WL:IF ST<LEN(AS)+1 GOTO 600
680 RETURN
800 IF LEN(AS)>LL THEN CLS:PRINT "ERRO NO FORMATO. NAO POSSO CENTRALIZAR",AS:END
810 PRINT #P,CHR$(13);
820 IF HW>LEN(AS)/2 THEN PRINT #P,STRING$(HW-LEN(AS)/2," ");
830 ST=2:RETURN
```



```
10 LET LL=32:LET PL=32
15 LET LL=LL+1:LET T=(PL-LL)/2
20 LET D=0
30 READ AS:LET L=LEN AS
40 LET C=0
50 IF C=L THEN GOTO 30
60 LET C=C+1:LET D=D+1:IF C>1 THEN GOTO 100
70 IF AS(C)="#" THEN GOTO 500
80 IF AS(C)="*" THEN GOTO 700
85 IF AS(C)="&" THEN GOTO 850
90 IF AS(C)="$" THEN LPRINT CHR$(13):LPRINT CHR$(13):LET D=D+1:GOTO 900
95 LET AS=" "+AS:LET L=L+1
100 IF MID$(AS,C,1)="#" THEN GOTO 800
110 LPRINT MID$(AS,C,1);
115 IF D>LL THEN LET D=0
120 GOTO 50
500 LET NL=0:LET TA=LL:LET BE=0
510 LET LE=LEN(AS)-1:IF LE>LL THEN
```

```
850
90 IF AS(C)="$" THEN LPRINT CHR$(13):CHR$(13):LET D=D+1:GOTO 900
95 LET AS=" "+AS:LET L=L+1
100 IF AS(C)="#" THEN GOTO 800
110 LPRINT AS(C)
115 IF D>LL THEN LET D=0
120 GOTO 50
500 LET NL=0:LET TA=LL:LET BE=0
510 LET LE=LEN AS-1:IF LE>LL THEN PRINT FLASH 1;"Erro no formato - Endereco muito grande.":STOP
520 IF LE>BE THEN LET BE=LE
530 LET NL=NL+1:READ AS:IF AS(1)="#" THEN GOTO 510
540 RESTORE 1000
550 LET TR=T+LL-BE:FOR G=1 TO NL:FOR H=1 TO TR:LPRINT " ";:NEXT H:READ AS:LPRINT AS(2 TO ):NEXT G
560 GOTO 30
700 LET TA=(LL-L)/2+T:IF TA<T THEN PRINT CHR$(13):PRINT FLASH 1;"Erro no formato - Nao posso centralizar.":STOP
710 LPRINT CHR$(13);:FOR n=1 TO ta:LPRINT " ";:NEXT n:LPRINT AS(2 TO L):GOTO 20
800 LET SL=LL-D-1:LET CC=C+1:LET X=1
810 IF CC=L THEN GOTO 825
820 IF AS(CC)<>" " THEN LET CC=CC+1:LET X=X+1:GOTO 810
825 IF X>LL THEN PRINT CHR$(13):PRINT FLASH 1;"Erro no Formato - Palavra muito grande.":STOP
830 IF SL>X THEN GOTO 110
850 LPRINT CHR$(13);:LET D=0
900 FOR B=1 TO T:LPRINT " ";:NEXT B:GOTO 50
```



```
5 CLS
10 LET LL=35:LET PL=40
15 LET LL=LL+1:LET PL=(PL-LL)/2
20 LET D=0
30 READ AS:LET L=LEN(AS)
40 LET C=0
50 IF C=L THEN GOTO 30
60 LET C=C+1:LET D=D+1:IF C>1 THEN GOTO 100
65 BS=MID$(AS,C,1)
66 IF BS="#" THEN STOP
70 IF BS="#" THEN GOTO 500
80 IF BS="*" THEN GOTO 700
85 IF BS="&" THEN GOTO 850
90 IF BS="$" THEN LPRINT CHR$(13):LPRINT CHR$(13):LET D=D+1:GOTO 900
95 LET AS=" "+AS:LET L=L+1
100 IF MID$(AS,C,1)="#" THEN GOTO 800
110 LPRINT MID$(AS,C,1);
115 IF D>LL THEN LET D=0
120 GOTO 50
500 LET NL=0:LET TA=LL:LET BE=0
510 LET LE=LEN(AS)-1:IF LE>LL THEN
```

```

HEN PRINT "ERRO NO FORMATO - En
dereço muito longo"
520 IF LE>BE THEN LET BE=LE
530 LET NL=NL+1:READ AS:IF MID$
(AS,C,1)="# THEN GOTO 510
540 RESTORE 1000
550 LET TR=T+LL-BE:FOR G=1 TO N
L:FOR H=1 TO TR:LPRINT " ";:NEX
T H:READ AS:F=LEN(AS)-1:LPRINT
RIGHTS(AS,F):NEXT G
560 GOTO 30
700 LET TA=(LL-L)/2+T:IF TACT T
HEN LPRINT CHR$(13):PRINT "ERRO
NO FORMATO - Não posso central
izar":STOP
710 LPRINT CHR$(13):FOR N=1 TO
TA:LPRINT " ";:NEXT N:F=LEN(AS)
-1:LPRINT RIGHTS(AS,F):GOTO 20
800 LET SL=LL-D-1:LET CC=C+1:LE
T X=1
810 IF CC=L THEN GOTO 825
820 IF MID$(AS,CC,1)<>" " THEN
LET CC=CC+1:LET X=X+1:GOTO 810
825 IF X>LL THEN LPRINT CHR$(1
3):PRINT "ERRO NO FORMATO - Pal
avra muito grande":STOP
830 IF SL>X THEN GOTO 110
850 LPRINT CHR$(13):LET D=0
900 FOR B=1 TO T:PRINT " ";:NEXT
B:GOTO 50

```



```

100 HOME
110 ONERR GOTO 430
120 TL = 80:LL = 60:P = 1:D$ =
CHR$(4)
130 SP = (TL - LL) / 2:HW = TL
/ 2
140 PRINT D$:"PR#":P
150 PRINT CHR$(13)
160 READ AS:AS = AS + CHR$(3
2)
170 IF AS = "" THEN 160
180 OS = LEFT$(AS,1)
190 IF OS = "#" THEN ML = 0: G
OSUB 260: GOTO 250
200 IF OS = "$" THEN RL = 0: P
RINT CHR$(13): CHR$(13): SPC
( SP)::ST = 2: GOTO 240
210 IF OS = "*" THEN GOSUB 40
0:RL = HW - LEN(AS) / 2: GOTO
240
220 IF OS = "&" THEN RL = 0: P
RINT CHR$(13): SPC( SP)::ST =
2: GOTO 240
230 ST = 1
240 GOSUB 310
250 GOTO 160
260 IF LEN(AS) > ML THEN ML
= LEN(AS)
270 N = N + 1: READ AS: IF LEF
T$(AS,1) = "#" THEN 260
280 IF ML > LL THEN PRINT D$:
"PR#0": HOME: PRINT: PRINT "E
RRO DE FORMATO! O ENDEREÇO": PR
INT "E MUITO LONGO PARA O TAMAN
HO DA LINHA": END
290 RESTORE: FOR J = 1 TO N:
READ AS: PRINT SPC( LL - ML +
SP + 1): RIGHTS(AS, LEN(AS) -
1): CHR$(13):
300 NEXT: RETURN

```

1000 DATA "Município de São Paulo"  
 1010 DATA "Rua Manoel de Almeida, 124"  
 1020 DATA "Indústria de Cimento, S/A"  
 1030 DATA "São Paulo"  
 1040 DATA "São Paulo"  
 1050 DATA "São Paulo"  
 1060 DATA "São Paulo"  
 1070 DATA "São Paulo"  
 1080 DATA "São Paulo"  
 1090 DATA "São Paulo"  
 1100 DATA "São Paulo"  
 1110 DATA "São Paulo"  
 1120 DATA "São Paulo"  
 1130 DATA "São Paulo"  
 1140 DATA "São Paulo"  
 1150 DATA "São Paulo"  
 1160 DATA "São Paulo"  
 1170 DATA "São Paulo"  
 1180 DATA "São Paulo"  
 1190 DATA "São Paulo"  
 1200 DATA "São Paulo"

Município de São Paulo  
 Rua Manoel de Almeida, 124  
 Indústria de Cimento, S/A

Município de São Paulo  
 Rua Manoel de Almeida, 124  
 Indústria de Cimento, S/A

Município de São Paulo

Município de São Paulo

Município de São Paulo  
 Rua Manoel de Almeida, 124  
 Indústria de Cimento, S/A

Município de São Paulo  
 Rua Manoel de Almeida, 124  
 Indústria de Cimento, S/A

Município de São Paulo  
 Rua Manoel de Almeida, 124  
 Indústria de Cimento, S/A

Município de São Paulo

Município de São Paulo

Município de São Paulo  
 Rua Manoel de Almeida, 124  
 Indústria de Cimento, S/A

```

310 WL = 0
320 IF ST + WL > LEN(AS) THE
N 340
330 IF MID$(AS,ST + WL,1) <
">" THEN WL = WL + 1: GOTO 3
20
340 IF WL > LL THEN PRINT D$:
"PR#0": HOME: PRINT "ERRO DE F
ORMATO! "; PRINT AS;"... CONTEM
UMA PALAVRA MAIOR QUE A LINHA!
": END
350 IF RL + WL - 1 > LL THEN
PRINT CHR$(13): SPC( SP)::RL
= 0
360 WL = WL + 1

```

```

370 PRINT MID$(AS,ST,WL)::RL
= RL + LEN(MID$(AS,ST,WL))
380 ST = ST + WL: IF ST < LEN
(AS) + 1 THEN 310
390 RETURN
400 IF LEN(AS) > LL THEN PR
INT D$:"PR#0": HOME: PRINT "ER
RO DE FORMATO! NAO SE PODE CENT
RALIZAR ";AS: END
410 PRINT CHR$(13):
420 IF HW > LEN(AS) / 2 THEN
PRINT SPC( HW - LEN(AS) /
2):ST = 2: RETURN
430 PRINT: PRINT D$:"PR#0": E
ND

```

LINHA	FABRICANTE	MODELO	FABRICANTE	MODELO	PAÍS	LINHA
Apple II +	Appletronica	Thor 2010	Appletronica	Thor 2010	Brasil	Apple II +
Apple II +	CCE	MC-4000 Exato	Apply	Apply 300	Brasil	Sinclair ZX-81
Apple II +	CPA	Absolutus	CCE	MC-4000 Exato	Brasil	Apple II +
Apple II +	CPA	Polaris	CPA	Absolutus	Brasil	Apple II +
Apple II +	Digitus	DGT-AP	CPA	Polaris	Brasil	Apple II +
Apple II +	Dismac	D-8100	Codimex	CS-6508	Brasil	TRS-Color
Apple II +	ENIAC	ENIAC II	Digitus	DGT-100	Brasil	TRS-80 Mod.III
Apple II +	Franklin	Franklin	Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Apple II +	Houston	Houston AP	Digitus	DGT-AP	Brasil	Apple II +
Apple II +	Magnex	DM II	Dismac	D-8000	Brasil	TRS-80 Mod. I
Apple II +	Maxitronica	MX-2001	Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Apple II +	Maxitronica	MX-48	Dismac	D-8100	Brasil	Apple II +
Apple II +	Maxitronica	MX-64	Dynacom	MX-1600	Brasil	TRS-Color
Apple II +	Maxitronica	Maxitronic I	ENIAC	ENIAC II	Brasil	Apple II +
Apple II +	Microcraft	Craft II Plus	Engebras	AS-1000	Brasil	Sinclair ZX-81
Apple II +	Milmar	Apple II Plus	Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Apple II +	Milmar	Apple Master	Franklin	Franklin	USA	Apple II +
Apple II +	Milmar	Apple Senior	Gradiente	Expert GPC1	Brasil	MSX
Apple II +	Omega	MC-400	Houston	Houston AP	Brasil	Apple II +
Apple II +	Polymax	Maxxi	Kemtron	Naja 800	Brasil	TRS-80 Mod.III
Apple II +	Polymax	Poly Plus	LNW	LNW-80	USA	TRS-80 Mod. I
Apple II +	Spectrum	Microengenho I	LZ	Color 64	Brasil	TRS-Color
Apple II +	Spectrum	Spectrum ed	Magnex	DM II	Brasil	Apple II +
Apple II +	Suporte	Venus II	Maxitronica	MX-2001	Brasil	Apple II +
Apple II +	Sycomig	SIC I	Maxitronica	MX-48	Brasil	Apple II +
Apple II +	Unitron	AP II	Maxitronica	MX-64	Brasil	Apple II +
Apple II +	Victor do Brasil	Elppa II Plus	Maxitronica	Maxitronic I	Brasil	Apple II +
Apple II +	Victor do Brasil	Elppa Jr.	Microcraft	Craft II Plus	Brasil	Apple II +
Apple IIe	Microcraft	Craft IIe	Microcraft	Craft IIe	Brasil	Apple IIe
Apple IIe	Microdigital	TK-3000 IIe	Microdigital	TK-3000 IIe	Brasil	Apple IIe
Apple IIe	Spectrum	Microengenho II	Microdigital	TK-82C	Brasil	Sinclair ZX-81
MSX	Gradiente	Expert GPC-1	Microdigital	TK-83	Brasil	Sinclair ZX-81
MSX	Sharp	Hotbit HB-8000	Microdigital	TK-85	Brasil	Sinclair ZX-81
Sinclair Spectrum	Microdigital	TK-90X	Microdigital	TK-90X	Brasil	Sinclair Spectrum
Sinclair Spectrum	Timex	Timex 2000	Microdigital	TKS-800	Brasil	TRS-Color
Sinclair ZX-81	Apply	Apply 300	Milmar	Apple II Plus	Brasil	Apple II +
Sinclair ZX-81	Engebras	AS-1000	Milmar	Apple Master	Brasil	Apple II +
Sinclair ZX-81	Filcres	NEZ-8000	Milmar	Apple Senior	Brasil	Apple II +
Sinclair ZX-81	Microdigital	TK-82C	Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Sinclair ZX-81	Microdigital	TK-83	Omega	MC-400	Brasil	Apple II +
Sinclair ZX-81	Microdigital	TK-85	Polymax	Maxxi	Brasil	Apple II +
Sinclair ZX-81	Prologica	CP-200	Polymax	Poly Plus	Brasil	Apple II +
Sinclair ZX-81	Ritas	Ringo R-470	Prologica	CP-200	Brasil	Sinclair ZX-81
Sinclair ZX-81	Timex	Timex 1000	Prologica	CP-300	Brasil	TRS-80 Mod.III
Sinclair ZX-81	Timex	Timex 1500	Prologica	CP-400	Brasil	TRS-Color
TRS-80 Mod. I	Dismac	D-8000	Prologica	CP-500	Brasil	TRS-80 Mod.III
TRS-80 Mod. I	Dismac	D-8001/2	Ritas	Ringo R-470	Brasil	Sinclair ZX-81
TRS-80 Mod. I	LNW	LNW-80	Sharp	Hotbit HB-8000	Brasil	MSX
TRS-80 Mod. I	Video Genie	Video Genie I	Spectrum	Microengenho I	Brasil	Apple II +
TRS-80 Mod.III	Digitus	DGT-100	Spectrum	Microengenho II	Brasil	Apple IIe
TRS-80 Mod.III	Digitus	DGT-1000	Spectrum	Spectrum ed	Brasil	Apple II +
TRS-80 Mod.III	Kemtron	Naja 800	Suporte	Venus II	Brasil	Apple II +
TRS-80 Mod.III	Prologica	CP-300	Sycomig	SIC I	Brasil	Apple II +
TRS-80 Mod.III	Prologica	CP-500	Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
TRS-80 Mod.III	Sysdata	Sysdata III	Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
TRS-80 Mod.III	Sysdata	Sysdata Jr.	Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
TRS-80 Mod.IV	Mullix	MX-Compacto	Timex	Timex 1000	USA	Sinclair ZX-81
TRS-80 Mod.IV	Sysdata	Sysdata IV	Timex	Timex 1500	USA	Sinclair ZX-81
TRS-Color	Codimex	CS-6508	Timex	Timex 2000	USA	Sinclair Spectrum
TRS-Color	Dynacom	MX-1600	Unitron	AP II	Brasil	Apple II +
TRS-Color	LZ	Color 64	Victor do Brasil	Elppa II Plus	Brasil	Apple II +
TRS-Color	Microdigital	TKS-800	Victor do Brasil	Elppa Jr.	Brasil	Apple II +
TRS-Color	Prologica	CP-400	Video Genie	Video Genie I	USA	TRS-80 Mod. I

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.

Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:



Sinclair ZX-81



TRS-80



TK-2000



MSX



Spectrum



TRS-Color



Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

# ■■■■■■■■■■ NO PRÓXIMO NÚMERO ■■■■■■■■■■

## CÓDIGO DE MÁQUINA

Sistemas numéricos. O sistema decimal e o sistema binário. Bits e Bytes. Como o computador faz contas de somar.

## PROGRAMAÇÃO DE JOGOS

Controle os movimentos da figura na tela. Jogos de guerra. Dispare seus mísseis e destrua o inimigo!

## PROGRAMAÇÃO BASIC

A arte de fazer laços FOR... NEXT. Aprenda a criar efeitos sonoros no computador e veja o pôr-do-sol.

CURSO PRÁTICO **2** DE PROGRAMAÇÃO DE COMPUTADORES

# INFORMÁTICA

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

